Spring 4-1-2008

Student Data Repository and Warehouse

Eric Urff
Dakota State University

Student Data Repository and Warehouse

A graduate project submitted to Dakota State University in partial fulfillment

of the requirements for the degree of

Master of Science

in

Information Systems

April, 2008

By

Eric Urff

Project Committee:

Dr. Stephen Krebsbach

Dr. Wayne Pauli

Dr. Mark Moran

PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Eric Urff

Master's Project Title: Student Data Repository and Warehouse

Faculty supervisor: _____ Date: 5/19/08

Committee member: _____ Date: 5/19/08

Committee member: _____ Date: 5/19/08

ACKNOWLEDGMENT

I would like to thank my coworkers and friends at Daytona Beach College that have helped me with details and lending an ear as this project unfolded. Without them it would not have taken shape nor been completed.

I also need to express my sincere love and gratitude to my loving wife Susie, who has seen me work through the MSIS program at DSU and supported me through thick and thin. My wife has supported me both directly and indirectly through the process of creating this work.

ABSTRACT

This paper is about the process of designing and creating a data repository/warehouse of student elements from a transactional student database. The original data exists in an Informix database that has millions of records within the data structures. The ability to extract data and report on necessary elements from the transactional database is a tedious and ominous task that can sometimes take lengthy amounts of time to obtain necessary information for key analysis.

The need to simplify the records and make the information available on a timely basis to users throughout the college is the main goal. By deciding to use a MYSQL database for the data repository/warehouse the system can easily be moved to any type of platform with minimal cost. By extracting and placing the data in another database it is now possible to run reports against data without affecting response times on the transactional system.

One of the key elements of extraction and loading of data was to use the HPUX platform for all phases. This allows the scripts to be automated within one system and has the ability to run efficiently. By experimenting with different data layouts and indexes the load times were drastically reduced with minimal effort. It is important to note that making sure that all of the data is clean before loading it into the new system or the reporting elements are useless to the college.

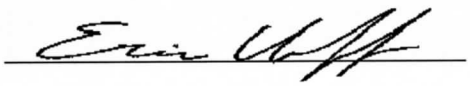
This is a project that appears to have actually just gotten off the ground and will have months if not years of further development. Getting started on a simple portion of what may prove to be an extremely large process proved to be the key to success. There is no doubt that the spiral methodology was the correct option to use during this project.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,



Eric Urff

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Background of the Problem

There is a need at Daytona Beach College (DBC) to be able to retrieve data in order to report and analyze information on different aspects of the students and classes. Presently requests for information are sent to Information Services in order to get the information that is desired. These reports are created and run "on demand" which can take several minutes up to an hour to get the necessary data back for a specific area of information that is requested.

All areas have the desire to know what the full time enrollment (FTE) counts are on a daily basis. Part of this is due to the reason that the funding for the college is driven by these numbers along with the fact that deans and faculty need to know how their areas of study are progressing. Additionally, all detailed information is extracted and submitted to the Department of Education at the state level at the beginning and end of each terms. This data is then interpreted, maintained, and used to drive the funding for the colleges. The information needs to be available to administrative personnel during the semester to determine what plans of action can be used to increase enrollment and retention.

Statement of the problem

The current data resides in a transactional database with over 750 tables and more than 50 million records of information. This includes over 600,000 students with all the associated records that go with a registered student. The college has courses from the high school level up through college level with over 200 different majors.

There are no methods for obtaining rolled up data to report the total FTE numbers and registrations to the departments and administration without having to run multiple reports, which take significant amounts of time. Reports for specific class information can be obtained through the current ERP system but this usually limits the information to what is already present on the report. If any changes need to be reflected then the programming staff needs to modify the report and place it back into production.

Some departments do have the capability to use a reporting package and create internal reports to obtain data from the system. This is both good and bad for the institution. To create a report a user must understand what information is stored in which tables and how to link that information together. There is currently no dictionary that has the links predefined. If a user creates a report and does not put in the correct selection statements or links tables that should not be linked then the data can be erroneous and nobody will know that it is wrong. There has been the case when three reports were written for the same information by three report writers resulting in three different sets of results. Time must then be spent to analyze each report and determine the correct outcome.

Objectives of the project

The data warehouse/repository database for the college will be created from extracting and combining data from the transactional system and restructured for reporting purposes. The need to get information in a timely fashion is the main goal of the project. The goal is to allow departments to obtain information that can be compared semester to semester, or year to year by departments, courses, students, etc.

Along with the reporting database multi-dimensional cubes will also be created in order to use the power of data extraction and manipulation with OLAP tools. All of the new reporting information will be made available via a web interface.

The deliverables will include reports that will be created that end users can run on demand. An extract and cleanup routine will be created to ensure that the data is clean that is input into the new database. As much as is possible the actual data in the transactional database will be cleaned and updated within the database to ensure that other reporting functions all access "clean" data for reporting purposes. The Cognos reporting suite of tools will be used to present data via a web interface with the necessary links to the database.

CHAPTER 2

LITERATURE REVIEW

Higher Educational institutions have a need for data that differs from a usual private sector environment. The three challenges according to Guan, Nunez and Walsh (2002), that colleges face are: (1) administration needs better access to data and information in order to make informed decisions, (2) increasing competition among institutions puts a demand for more information to help with recruiting and retention of students, and (3) the demand from both state and federal agencies are requiring more information from colleges in order to track their performance. Colleges have a myriad of data available in transactional systems. However data is rarely organized in a manner in which department heads and administrators can readily access on a daily basis.

Within the state of Florida the Department of Education (DOE), a committee known as MISATFOR (Management Information Systems Advisory Task Force) which contains members from the 28 Community Colleges throughout the state. This committee has strict standards and guidelines that dictate what data is maintained and tracked for each college. All of the colleges extract and transmit data to five separate databases:

- Student Data Base
- Personnel Data Base
- Annual Personnel Reports Database
- Facilities & Capital Outlay Database
- Integrated Data Base

- Admissions Data Base

The data from each college is normally extracted at the beginning and ending of each term from each institution's transactional systems and then submitted to the state. Reports are retrieved and run from these submissions to validate the data. Each college then gets the "official" enrollment information that state funding will be based on.

Many different approaches to system development are available for creating and implementing computer systems. The waterfall process incorporates a logical flow from the planning phase through the implementation phase. According to Giovinazzo (2000) this approach usually requires a defined set of specific requirements. The data warehouse will not lend itself to a specific set of standards since the data will be in continuous flux and the reports and information that will be extracted will be changing continuously. The process that will be followed overall for this project will be the spiral design which incorporates the following phases: Planning, Design, Implementation, and Evaluation. Nilesh (2005) describes the spiral model as a way to eliminate some of the risks that are incorporated in the waterfall process. As can be seen in the diagram below with this method we will start with the planning phase and progress through the processes. The system will start at the center and will have a never-ending outward spiral that allows for continuous redevelopment and additional information as needed.

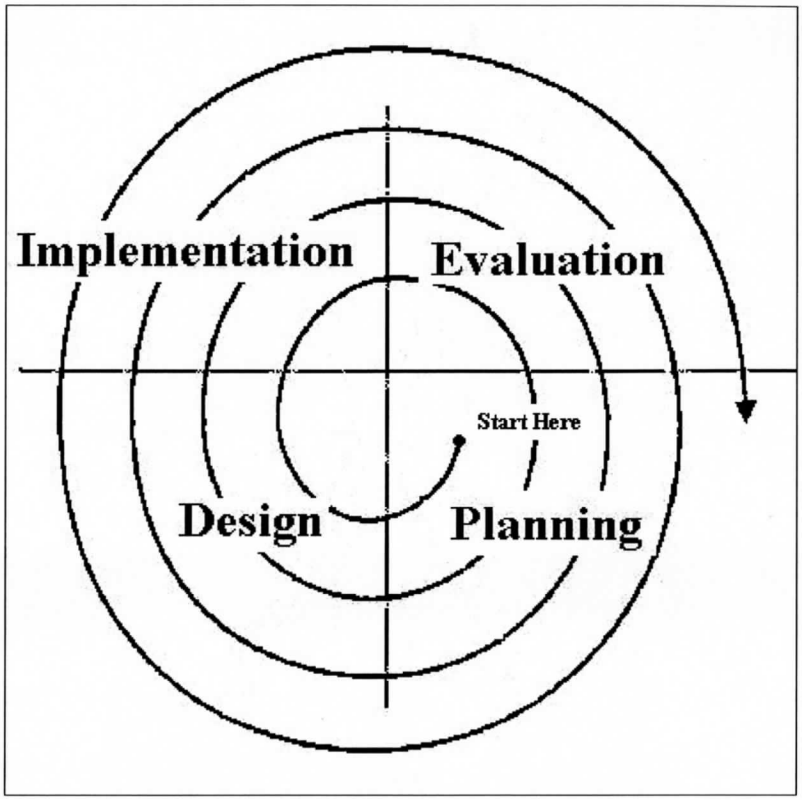


Figure 4.1 Spiral Methodology Diagram

The solution for DBC is to extract data from the transactional database tables along with any data cleansing that may be necessary and build a database that can be accessed for information in a timely manner. The initial phase will be to create data that can identify registration counts and FTE (full time equivalency) on a daily basis, broken down into different divisions, departments, courses, etc. This will just be the first phase of an ongoing project that will continue on for an indefinite time. The data requirements will be in constant flux and changing with each loop through the spiral methodology defined above. This

project will be deemed a success when the daily registration data is made available via the web to all administrative units of the college.

CHAPTER 3

SYSTEM DESIGN

This first phase of the data information system will be to determine daily reporting information for DBC. The phases will follow the basic layout of the spiral method which will include: planning, design, implementation, and evaluation phases. The individual phases will be broken down further as the project progresses through the first spiral.

Preliminary Information

Information that will be accessed and consolidated will arise from the transactional database that is run at the college. At current there are well over 700 tables containing approximately 30GB of data that resides in the tables. The data contained in the database is maintained in 2NF and 3NF form with the appropriate links that need to be understood in order to obtain the correct relationships for reporting purposes. This includes some outer joins (which can lead to inefficient queries) due to the lack of specific links and possible data inconsistencies. This information will need to be put together in a format that can easily be retrieved for analysis purposes.

The college has purchased the Cognos suite of reporting tools that can access databases through ODBC connections. This package is also capable of creating OLAP cubes for analysis functions. These will be what some of the final reporting elements are desired to be run against. The Informix database has all of the transactional data and will require

extraction and cleansing of the data prior to loading into the data informational system. MYSQL has been chosen to load the data into since it is an open source product and can run on the same platform as the Informix database (HPUX). Other programming tools that are currently available at the college and will be used as necessary include: Borland C++ Builder, Informix tools (ISQL, ACE reporting), and all commands on the HPUX system.

The MYSQL platform is used in order to reduce the need for backups on a daily basis. Since the MYSQL database of information will be very large and in order to improve response time without directly affecting the transactional system it will be separated from the transactional database. If desired in the future the actual warehousing tables can be put into the same database with minimal changes.

Planning Phase

The key stakeholders of this project have been identified as the Dean of Records and Registration along with the Vice President of Enrollment Services. They each have a vested interest in obtaining information on a daily basis that can be compared to previous years and semesters. This will allow them to determine registration trends and patterns so that retention and enrollment can be monitored. Department chairs will also have a need for the data so that scheduling of classes and sections can be set up to accommodate for any increases or decreases in enrollment that may take place over the registration period.

The initial information has been determined to be an extract of the PROGRAM, COURSE_CATEGORY, and CAMPUS breakdowns of the FTE and HEADCOUNT on a daily basis (see Table 5.1). This data will need to be available so that users can compare day to day from the previous year to the current year for analysis.

Table 5.1 Daily reporting requirement worksheet.

	Current Year Headcount	Prior Year Headcount	% Chg	Current Year FTE	Prior Year FTE	% Chg
Program						
College Credit						
AA						
AAS						
PSV						
BS Educ						
Apprentice						
CPRP						
PSAV						
VPRP						
Dual						
Total						
Total Unduplicated						
Adult Education						
ABE						
AHS						
ATC						
ESL						
GED						
Dual						
Total						
Total Unduplicated						
DBCC TOTAL						
Daytona Total						
Daytona CC						
Daytona AE						

The main source of data for this information is currently contained within the CRS_REC, CW_REC, SEC_REC, ID_REC, PROG_ENR_REC and STU_ACAD_REC. This is just the basis for the main data is does not include the linkages to the detailed informational tables that will need to be extracted. The corresponding tables and links are shown in Figure 5.2. It will be the extraction routine that will be required to denormalize this information and put into the tables that will be necessary for the data reporting tables to function.

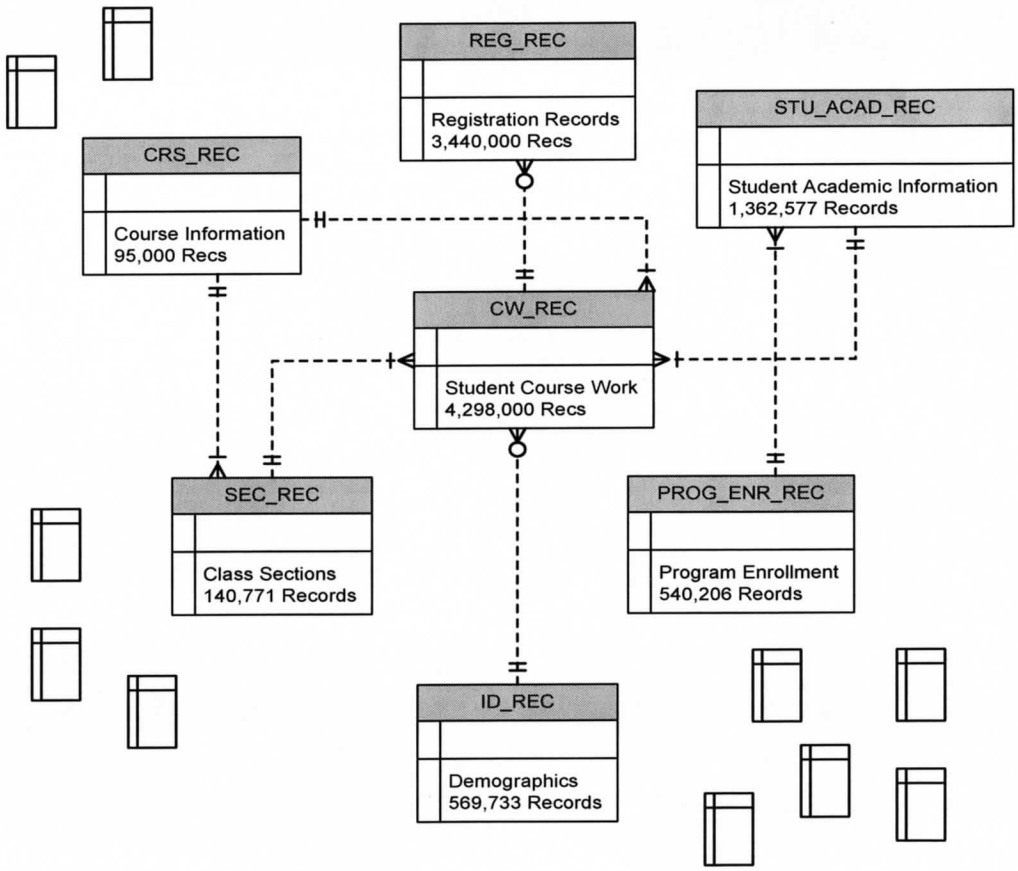


Figure 5.1 Transactional Data Tables (Main Information Subset).

Extra variables will be extracted that will also match up with the reporting requirements from the MISATFOR state databases. Cip codes for each course will be extracted so that preliminary information will be available to be analyzed early in each semester prior to extractions and submissions of the official data to the state system.

Design Phase

This project is very dependent on the creation and layout of the tables in the new database. The database will be put more toward 1NF so that data for reporting will be able to be reported on in a timely fashion without the need of determining all of the necessary links between tables, and having to worry about outer joins on tables.

The database will be designed with a star schema plan in mind. This is the preferred method for access using the Cognos reporting tools and putting the data in this format will be easily understood by users of the system. These tables have been de-normalized so that information is readily available. The information will be accumulated in the data warehouse database as follows:

- REG_DETAIL_FACTS – Contains the registration details for each student that occurred on each day. Note that numbers could become negative registration information in the event of a drop. This will be necessary to keep the totals balanced out to the actual counts at the end of each semester. In order to obtain the total registrations for a semester using this fact table it will be necessary to accumulate all prior records (by time_key) for a particular term.

- REGISTRATION_FACTS – Contains information about the totals of registrations, fte, and hours for each section (CLASS_TABLE) for each particular day of the year. Records will NOT need to be accumulated to get the total information for a particular day.
- TIME_DIMENSION – Contain a time_key (date) for each day of the year. This will be linked to the FACTS tables for comparison purposes and specific information based upon each day. The specific number of days prior and after the start of each semester will be calculated and stored. This is the usual means of comparisons from one year to the next. There will also be similar date information kept for the accounting date periods for a future FACTS table that will store accounting information.
- CLASS_DETAILS – Each section of each course will be maintained here. Each record will be larger than the standard tables in the transactional system since it will be a combination of information.
- TERM_TABLE – Term specific information will be maintained here to have a link back to what records belong to what terms.
- DEMOGRAPHICS – Demographical information for each student in the system.
- DEPT_TABLE - Department information that courses relate to.
- DIV_TABLE – Divisions that maintain the department information.

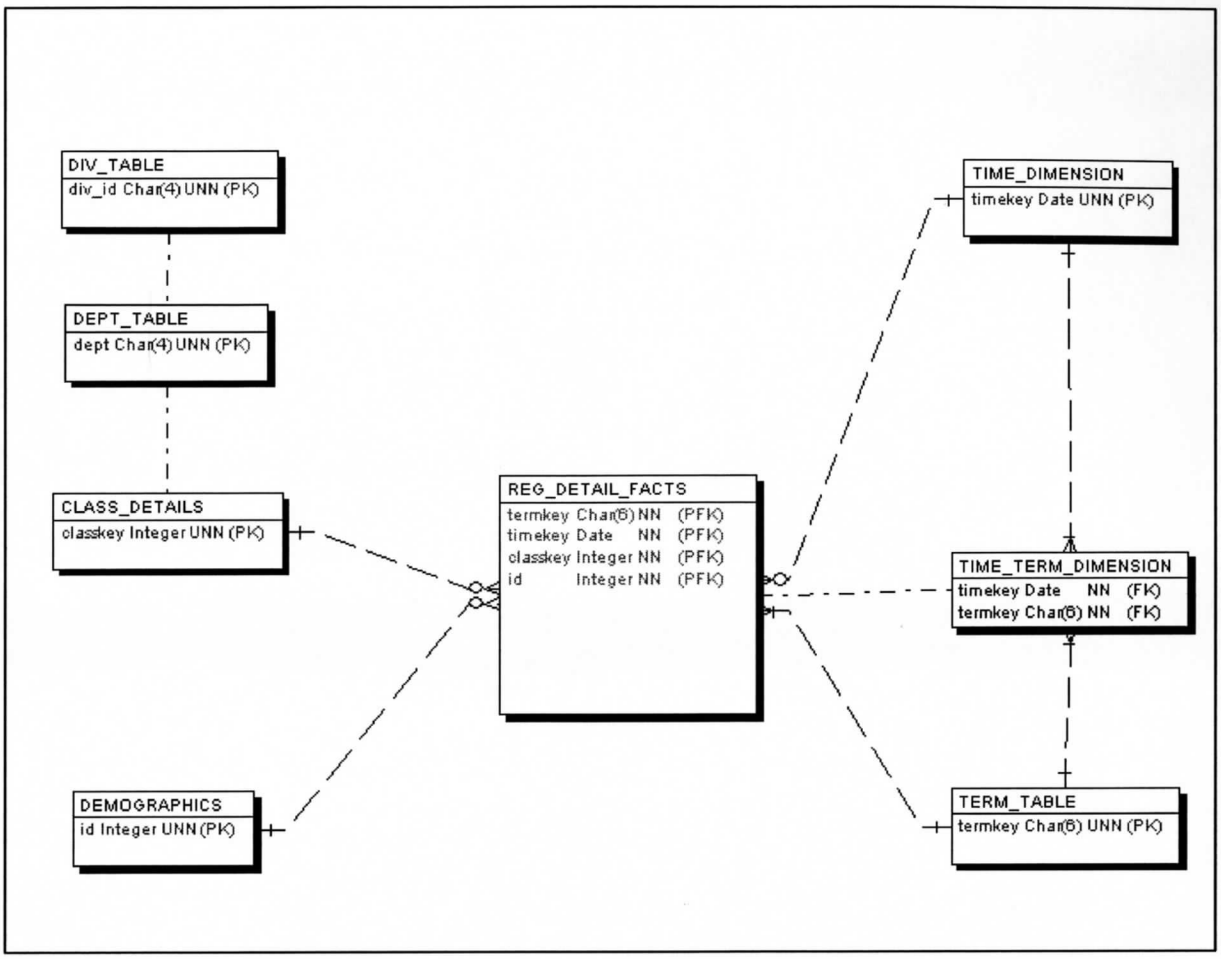


Figure 5.2 Star Schema Diagram using REG_DETAIL_FACTS table.

The decision and design of the attributes for the tables is very similar to the transactional database information. All of the attributes for each table have been determined and have been labeled with descriptions of what information is expected to be found in them.

The time dimension table includes actual counts of the number of days between the time_key and the start of classes for the corresponding term. In order to eliminate redundancy and confusion the decision has been made to track individual days from five months previous to the start of a semester up until 7 months after the start of a semester. Each of the major

semesters will have this calculation of the required value determined and loaded into the time dimension table.

The two different facts tables will be distinct to each other in that the REG_DETAIL_FACTS will be the actual registration totals by student for each day for each class. The REGISTRATION_FACTS table will be an accumulated total of information for each class in the corresponding semester. The difference here is that for a selected day in order to get the total registration information up until that point in time for the specified semester the REGISTRATION_FACTS will only need to go to the specific date in the table to accumulate the information, the REG_DETAIL_FACTS will need to be an accumulation of all days prior to and including the specific date to obtain the same number.

The remaining tables are informational and are accumulated by de-normalizing the transactional database information into the specific attributes as necessary. Extra attributes may need to be added in the future for other reporting needs, but at this first pass through the spiral a decision has been determined to produce output that will be useful to the college community.

Table 5.2. CLASS_DETAILS Attributes.

Key	Attribute/role name	Data type	Not null	Unique	Comments
PK	classkey	Integer	YES	YES	A unique key that links the information to the REGISTRATION_FACTS table.
	crs_no	Char	YES	NO	The course identifier to distinguish this course.
	sec_no	Char	YES	NO	Section number of this specific course.
	cat	Char	YES	NO	Catalog that this course belongs to. (CC07, CC08, etc).
	yr	Integer	YES	NO	Year course is taught (4 digit year).
	sess	Char	YES	NO	Session in which this course is taught. (FA, SP, SU).
	subsess	Char	YES	NO	Subsession associated with this offering of course.
	course_title	Varchar	NO	NO	Title of course (using only the title1 field from the crs_rec).
	prog	Char	NO	NO	Program code. (CC, CE, AE).
	cip_no	Char	NO	NO	CIP Number that is related to the course offering.
	cip_description	Varchar	NO	NO	The CIP description as related to the CIP Number (cip_no).
	crsctgry	Char	NO	NO	Associated course category to CIP number.
	discipline	Varchar	NO	NO	The discipline to report this course in. i.e. College Credit, Adult Education, etc.
	dept	Char	NO	NO	Department that is responsible for this course.
	dept_descr	Varchar	NO	NO	Department Description.
	hrs	Double	NO	NO	Credit hours for specified section.
	clock_hrs	Double	NO	NO	Clock hours (sometimes referred to as the actual contact time spent in class for the term).
	fte_type	Char	NO	NO	Fte type that this course is. Used to calculate the actual FTE for each student. i.e. fte30=> hrs / 30 fte900=> clock_hours / 900
	tuit_code	Char	NO	NO	Tuition code for accounting/billing.
	fee_code	Char	NO	NO	Fee code for accounting/billing.
	bill_code	Char	NO	NO	Bill code for accounting/billing.
	gordon	Integer	NO	NO	Does the gordon rule apply to this section?
	fund	Char	NO	NO	Accounting Fund number.
	subfund	Char	NO	NO	Accounting Subfund Number.
	func	Char	NO	NO	Accounting Function Number.
	crs_major	Char	NO	NO	Course reporting major, that this course is associated with.
	crs_major_desc	Varchar	NO	NO	The description of the major that this course belongs to.
	div_id	Char	NO	NO	Division identifier.
	div_desc	Varchar	NO	NO	Division Description.
	max_reg	Integer	NO	NO	Maximum registrations allowed for this class.
	faculty_id	Integer	NO	NO	Id of main faculty person that is assigned to this class.
	faculty_name	Varchar	NO	NO	Faculty name assigned to this class.
	campus_code	Char	NO	NO	Campus identification code.
	campus_name	Varchar	NO	NO	Campus name where course is taught at.
	im	Char	NO	NO	Instructional Method.
	im_descr	Varchar	NO	NO	Instructional Method Description.
	dayeve_wkend	Varchar	NO	NO	Is this a day, evening, or weekend class?
	days	Char	NO	NO	Days of week the course is taught.
	start_time	Integer	NO	NO	Actual starting time of the class from the main meeting record.

Table 5.3. DEMOGRAPHICS Attributes.

Demographical information for each student in the system.

Key	Attribute/role name	Data type	Not null	Unique	Comments
PK	id	Integer	YES	YES	ID number that uniquely identifies each student within the system.
	fullname	Char	NO	NO	Fullname in CX format.
	firstname	Varchar	NO	NO	First Name
	lastname	Varchar	NO	NO	Last Name.
	addr_line1	Varchar	NO	NO	Address Line 1.
	addr_line2	Varchar	NO	NO	Address Line 2.
	city	Char	NO	NO	City.
	st	Char	NO	NO	State
	zip	Char	NO	NO	Zip code.
	county	Char	NO	NO	County of residency.
	phone	Varchar	NO	NO	Phone Number of individual.
	ethnic_code	Char	NO	NO	Ethnicity code.
	ethnicity	Varchar	NO	NO	Ethnicity.
	birth_date	Date	NO	NO	Date of birth.
	visa_code	Char	NO	NO	Visa code.
	visa_desc	Varchar	NO	NO	Visa description.
	gender	Char	NO	NO	Gender M/F

Table 5.4. DEPT_TABLE Attributes.

Department information.

Key	Attribute/role name	Data type	Not null	Unique	Comments
PK	dept	Char	YES	YES	Department identifier.
	dept_descr	Varchar	YES	NO	Department Description.
	dept_chair_id	Integer	YES	NO	Chairman id number that is in charge of this department.
	dept_chair_name	Varchar	NO	NO	Chairman fullname in charge of this department.
	div	Char	YES	NO	Division that the department belongs to.

Table 5.5. DIV_TABLE Attributes.

Division information for each department.

Key	Attribute/role name	Data type	Not null	Unique	Comments
PK	div	Char	YES	YES	Division identifier.
	div_descr	Varchar	YES	NO	Division Description.
	div_dean_id	Integer	YES	NO	Dean id number that is in charge of this division.
	div_dean_name	Varchar	NO	NO	Dean fullname in charge of this division.

Table 5.6. REG_DETAIL_FACTS Attributes.

Facts containing registration details.

Key	Attribute/role name	Data type	Not null	Unique	Comments
PFK	termkey	Char	YES	NO	
PFK	timekey	Date	YES	NO	
PFK	classkey	Integer	YES	NO	
PFK	id	Integer	YES	NO	
	prog	Char	NO	NO	Program that the particular course belongs to (AE, CC, CE).
	registrations	Integer	NO	NO	Number of registrations in this particular class/section on the specified day.
	credit_hours	Double	NO	NO	Credit Hours for the field.
	clock_hours	Double	NO	NO	Clock Hours - usually used for AE since it is physical class contact hours that are used for FTE purposes.
	fte	Double	NO	NO	FTE generated by this section. Usually calculated as (credit_hours/30) except for AE which is (clock_hours/900)

.

Table 5.7. REGISTRATION_FACTS Attributes.

Facts accumulated per class.

Key	Attribute/role name	Data type	Not null	Unique	Comments
PFK	termkey	Char	YES	NO	
PFK	timekey	Date	YES	NO	
PFK	classkey	Integer	YES	NO	
	registrations	Integer	NO	NO	Number of registrations in this particular class/section on the specified day.
	prog	Char	NO	NO	Program that the particular course belongs to (AE, CC, CE).
	credit_hours	Double	NO	NO	Credit Hours for the field.
	clock_hours	Double	NO	NO	Clock Hours - usually used for AE since it is physical class contact hours that are used for FTE purposes.
	fte	Double	NO	NO	FTE generated by this section. Usually calculated as (credit_hours/30) except for AE which is (clock_hours/900)

Table 5.8. TERM_TABLE Attributes.

Term specific information – one record per session.

Key	Attribute/role name	Data type	Not null	Unique	Comments
PK	termkey	Char	YES	YES	Term Key created to uniquely identify each term and session in from the FACTS tables. It will be a combination of yr and sess. Ex. 2007FA, 2008SP, etc
	sess	Char	YES	NO	Session.
	yr	Integer	YES	NO	Year of specific term.
	term_start_date	Date	YES	NO	Starting date of term. This is the value that will be used for all courses in a semester regardless of subsession(s) and/or programs. This table will only contain ONE sess/yr entry for each term.
	fscl_yr	Char	YES	NO	Accounting G/L year corresponding to specific term.
	report_year	Char	YES	NO	The reporting year for the semester to appear in for college reporting purposes. Usually put in as 2007-2008, which is the academic year that the college wants most reports done as.
	report_sortorder	Integer	YES	NO	Sorting order key for the reporting year. Used to ensure that the results show up as FA, SP, SU
	state_reporting_year	Integer	YES	NO	The reporting year for the semester to appear in for STATE reporting purposes. i.e. 2007 would contain: SU 2007, FA 2007, SP 2008
	state_reporting_sortorder	Integer	YES	NO	Sorting order key for the reporting year. Used to ensure that the results show up as SU, FA, SP

Table 5.9 TIME_DIMENSION Attributes.

Time Dimension information for reporting purposes.

Key	Attribute/role name	Data type	Not null	Unique	Comments
PK	timekey	Date	YES	YES	Time key is the date in question. Format (YYYY-MM-DD)
	fscl_yr	Char	YES	NO	Fiscal year (0506, 0607, etc).
	fscl_start_date	Date	YES	NO	Starting date of fiscal year. This is used to determine the offsets for comparing days on a year to year basis. (Example: 0607=2006-06-30, 0708=2007-0701)
	fscl_first_monday	Date	YES	NO	This is the date that should reflect the first monday of the fiscal year. It could be from the preceding fiscal year (i.e. 0809 date would be 2008-06-30). This is used in order to compare like days (M-F) for accounting reports where assets need to be compared year to year, by actual day of week.
	fscl_days_from_start	Integer	YES	NO	The number of days from the start of the fiscal year to the time_key.
	fscl_month	Varchar	YES	NO	Fiscal Month.
	weekend_flag	Integer	YES	NO	Set to 1 if this day is a weekend, otherwise 0.
	holiday_flag	Integer	YES	NO	Set to 1 if a holiday, 0 otherwise.

Table 5.10 TIME_TERM_DIMENSION Attributes.

Time Term Dimension information. Keeps specific count of days prior/after start of term.

Key	Attribute/role name	Data type	Not null	Unique	Comments
FK	timekey	Date	YES	NO	
FK	termkey	Char	YES	NO	
	days_to_start	Integer	NO	NO	The number of days to the start of classes.

Development and Implementation Phase

The development phase of this project has taken many twists and turns in order to create the necessary tables and extraction of data for input into the warehouse. The table generation SQL statements are directly linked to MYSQL for creation. All extract and load statements have been put into the necessary formats and will be run in command line mode, so that they are easily reproducible for other institutions to manipulate to fit their desired needs. All necessary SQL statements for creation of the database are available in appendix C.

The MYSQL database was designed with INNODB table design. This was chosen with the main goal of having performance speed for the reporting aspects of the data. The fact tables have been designed with the main elements all being included as part of the primary key. This allows the most efficient and fast access to the data that will be retrieved on a daily basis. Figure 5.3 shows the relative retrieval times using the different tested indexes against the REG_DETAIL_FACTS table.

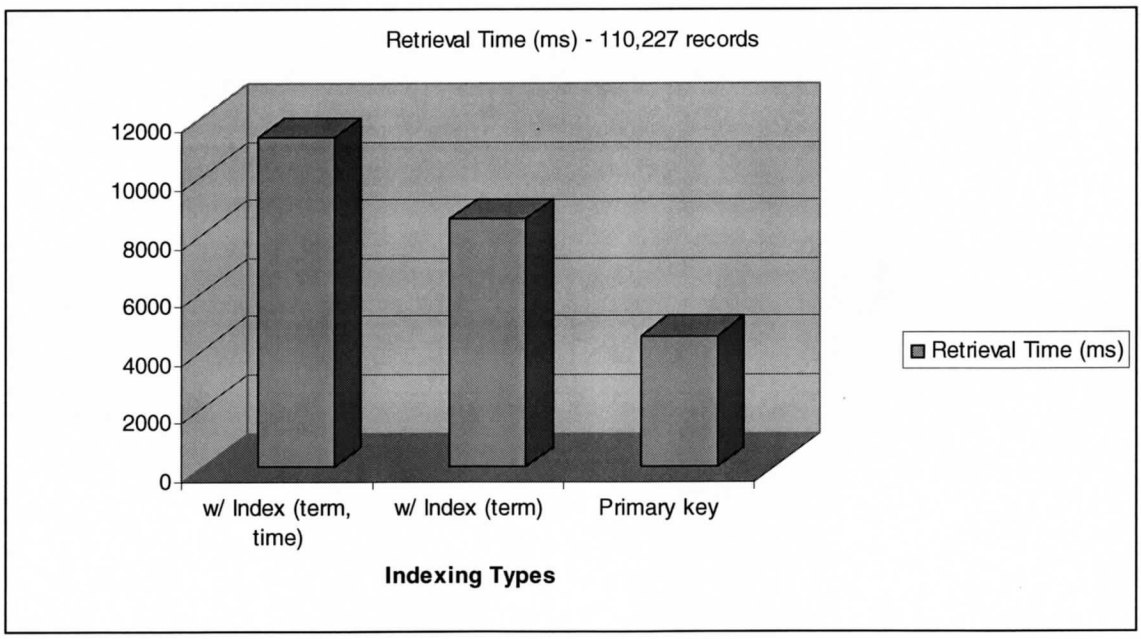


Figure 5.3 Retrieval Times for various indexing – REG_DETAIL_FACTS.

Dimension Table Extraction

The data extraction routines were developed for each table within the system. The data within the transactional database for DBC is relatively clean since all data is reported to the state. This decision to keep the data clean in the transactional database is critical to the data that is used for the warehouse. There are multiple reports that are run on a routine basis by different users of the college to ensure that the data is correct and within the proper guidelines for reporting purposes. When the system was originally migrated in 1996 there were many areas of the data that were not "clean" and were addressed at that time, for example: there was an instance of one high school that had 12 different records for the same school within the transactional database. Training is an ongoing project at this institution to ensure that data is input in a proper and consistent fashion. It cannot be stressed enough how important it is that the data in a system be kept as clean as possible to eliminate problems that can become increasingly large in the future.

The CLASS_DETAILS records consist of multiple extractions from the transactional database. Prior to running these extractions, a procedure has been created within the Informix (transactional) database to generate and maintain a distinct **classkey** that is used as the primary key for the table and links to the table. The data is denormalized into this one table for reporting purposes. The data is retrieved from about 10 tables from within the transactional system and combined together into an extraction file for purposes of importing into the data warehouse. The goal with this table and with most of the dimension tables in the new system is to have the data readily available and easy to access. Extraction and load statements are available in appendix C.

The time dimension was built with an entry for every day from 07/01/2005 through 6/30/2009. This table was built to include the first Monday of each fiscal year so that a comparison could be made from on fiscal year to the next based upon the actual day of week that revenue was generated. A field containing the number of fiscal days from the start of the fiscal calendar is used for comparison purposes. The fiscal fields are going to be used in subsequent phases of the warehousing data when financial records are created into their corresponding fact tables. A holiday flag was set to 0 (false) or 1(true) based upon the entries that exits in the HOLIDAY_TABLE in the transactional database. A C++ program was used to facilitate the necessary entries into this table for each day that was generated.

A TERM_TABLE was built which is an alternative type of time dimension for this reporting project. Most of all of the data that will be extracted will be term specific and a termkey has been created that will group all of the facts together in such a fashion, i.e. FA 2007 (fall semester, 2007) will become a key of 2007FA. This will allow quick and easy access by the keys necessary for reporting. The term table also includes key fields that are used throughout the institution and by the Florida State Reporting system. These fields include state report year and state reporting sort order, along with a report_year and report sort order. This is due to the fact that most reports that are desired by the institution count a year for enrollment purposes to be Fall, Spring, and Summer; while the State reporting semesters normally include Summer, Fall, and Spring. By developing the fields directly in the table there will be better access to the information on a consistent basis.

An alternative time dimension table (TIME_TERM_DIMENSION) has been built (see Appendix C) to facilitate the days prior to and after the starting date of a term. The desire by college administrators is to know the registration information 5 days prior to start of classes

for the current semester, compared to 5 days prior to start of classes for the same semester the previous year. This table makes a simple lookup possible to determine the exact date that is desired for the selection. A procedure was created in MYSQL that can be executed directly within a syntax so that the data may be looked up in one statement. The entire table contains 12240 records and only takes up 175K of disk space; this is a good trade for the speed of retrieval that we have obtained.

Facts Table Extraction

The goal is to extract the information for each registration on a daily basis. In the transactional based system (TBS) all of the registration details reside in a table known as the reg_rec. This table contains an entry for each individual of each class and whether or not the student Registered (R, r), Dropped (D,d), Cancelled (X, x), or Withdrawn (W,w) for each class. Assuming a value of +1 for each registration, -1 for each drop or cancellation, otherwise we assign a value of 0 to the record. When adding up the simple values of plus or minus one for each record then the final result should be equal to the corresponding value in the course work record for the student and class (cw_rec). The cw_rec is the official record of whether a student is counted as being enrolled in a class, if the cw_rec.stat is (R, r, W, w) then the student is counted as having been enrolled in the class and is counted for reporting purposes.

It is imperative to balance out the totals of each reg_rec to the final resulting value in the cw_rec before proceeding to the extraction phase of the registration_details_facts. Table 5.11 shows sample data that is in error.

Table 5.11 Erroneous Fact Table Data

retrieval order	cw_no	sys_date	tm	reg_stat	regval	running value
1	4132779	04/03/2007	2310	r	1	1
2	4132779	08/29/2007	2040	w	0	1
3	4132779	09/04/2007	1047	R	1	2
4	4132779	09/04/2007	1048	D	-1	1
	cw_rec.stat = D (implies total should be 0)					

Note that the running values start at 1 and proceed up to a value of 2 before going back down to a value of 1. The highlighted rows above indicate possible extraction errors that

may occur. The expected result should be a zero since the status of the course work record is that of a drop, which means that the student is no longer registered for the course and has dropped it. The sample script to test for these errors is included in Appendix C. In order to fix this problem we need to not select the 3rd record, this can be done by either eliminating the record (not a good idea) or setting a field within the record so that it is excluded from our search criteria. The latter choice is what has been chosen to do. A procedure is created to automatically locate the abnormalities for a session and term and attempt to fix the records so that these errors are no longer retrieved. After adjusting the records and running a subsequent retrieval of the same corresponding records the following information is obtained.

Table 5.12 Corrected registration details.

retrieval order	cw_no	sys_date	tm	reg_stat	regval	running value
1	4132779	04/03/2007	2310	r	1	1
2	4132779	09/04/2007	1048	D	-1	0
	cw_rec.stat = D (implies total should be 0)					

The data in table 5.12 now shows that the student was registered in the class on 04/03/2007 and would then count as being registered for our data/reporting purposes up until 09/04/2007 which is when the student subsequently drops the class. The retrieval of registration details will occur for a daily basis and will only include the total amount for a particular day for a student and corresponding class.

The data will also be assumed to be included to the preceding day if the actual registration record was created before 4am. This will allow for overnight processes to be completed which may affect some of the registration information. By allowing the above manipulations to take place and using appropriate techniques to balance out the data this is all

used to make sure that the data stays in balance and can be used after extraction. There will be the ability to pick any day of the year and see exactly what the registration totals were on that particular day by class, individucal, program, and many other classifications.

The actual data correction, extraction, and load routines are available in Appendix C. It should be noted that it is very efficient to extract the data in the same order as the primary key on the fact table. By performing this method the load time for Spring 2008 term wase reduced from over 5 minutes to 11 seconds. Testing different methods has proved extremely beneficial in the creation of the extract and load routines.

Reporting Phase

Once the data has been extracted and loaded into the warehouse being able to create reports and extractions is now available. With the new database extracted and loaded with data it is now possible to extract information in an efficient and expedient manner. Figures 5.4 and 5.5 show queries that have been developed to show the number of registrations by program and time of day for all registrations prior to 2/8/2008 for the spring term of 2008., the result set is included in figure 5.4. Both queries work to produce the same results but it takes 75-90 seconds to extract the data from the transactional database while the new warehouse can now extract the same information in less than 3 seconds. It is also a much simpler task to create the necessary sql in the new warehouse that has de-normalized tables than from the transactional database. It will take someone with load of experience to be able to create a sql similar to that in figure 5.4 versus the sql that is used in figure 5.5.

```
select  c.prog,
  case  when days like ('S------') then 'Weekend'
        when days like ('------S') then 'Weekend'
        when beg_tm <= 1700       then 'Day'
        when beg_tm >  1700       then "Evening"
         else 'Unknown'
  end
  , sum (case  when r.user_id < 0 then  0
               when r.stat in ('r', 'R') then  1
               when r.stat in ('d', 'D', 'x', 'X') then -1
               else 0
       end) as regval
  from reg_rec r, cw_rec c, sec_rec s
  ,secmtgmain_vw s1 ,    mtg_rec m, im_table imt
   where r.cw_no = c.cw_no
    and c.sess='SP' and c.yr=2008
    and sys_date < '2/8/2008'
   and s.yr=c.yr and s.sec_no=c.sec and s.sess=c.sess
   and  s.crs_no=c.crs_no
  and s1.crs_no = s.crs_no
  and s1.cat    = s.cat
  and s1.yr     = s.yr
  and s1.sess   = s.sess
  and s1.sec_no = s.sec_no
  and m.mtg_no  = s1.mtg_no
  and m.im      = imt.im
     group by 1,2
     order by 1,2
```

Figure 5.4 Retrieval Sample from transactional database (75 seconds).

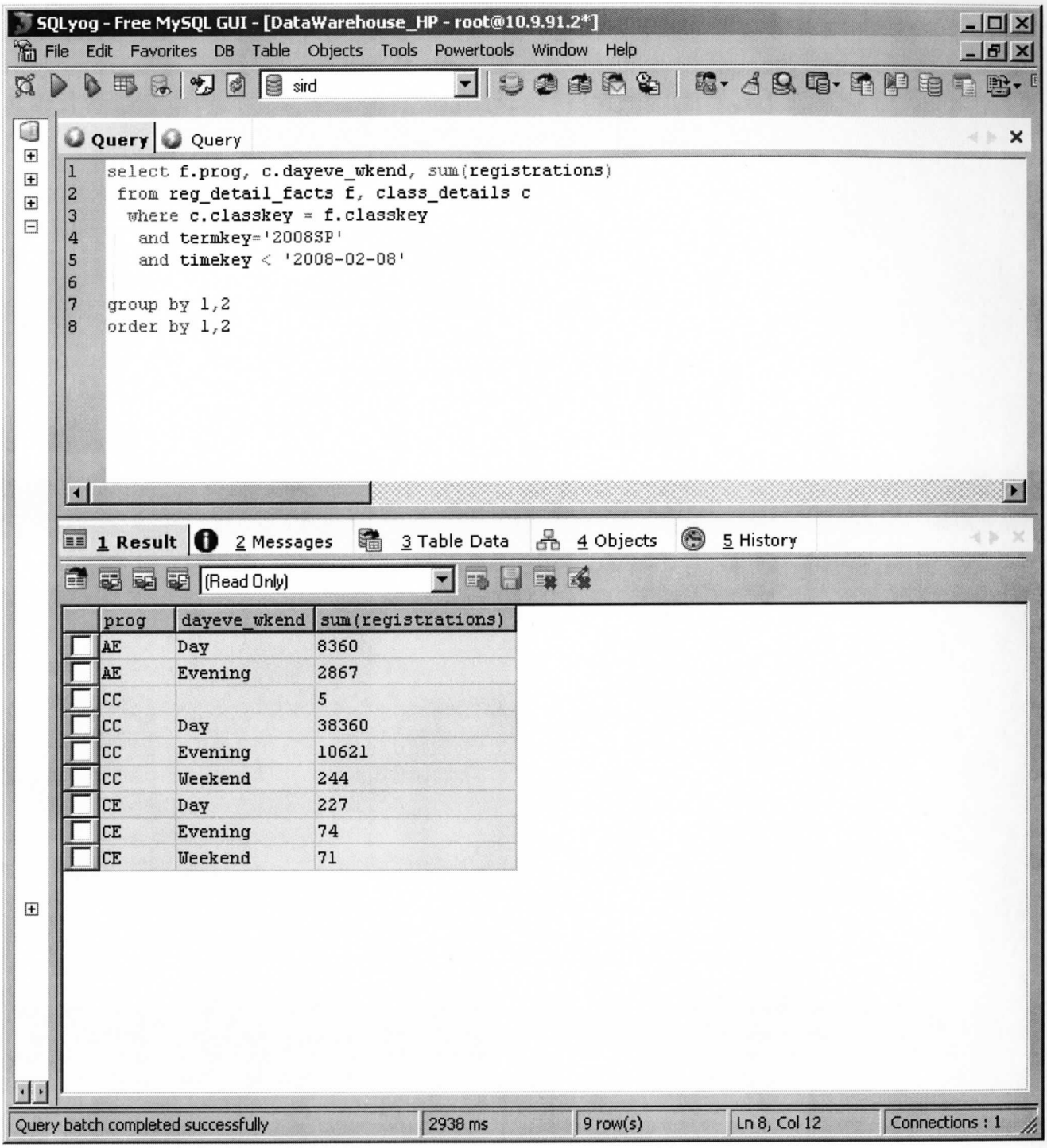


Figure 5.5 Retrieval Sample & results from extracted database (2.9 seconds).

Through the use of the cognos reporting tool we are now also able to create new reports and information in a matter of minutes to get meaningful information. The first step is to create the framework around which the reporting tool works. The process is to retrieve the tables and create the necessary links that end users will be able to use for creation of reports through the online reporting suite. Figure 5.6 shows the framework design of the tables and layouts, this is a very similar layout to the actual tables that were extracted in the development and extraction phase of the project.

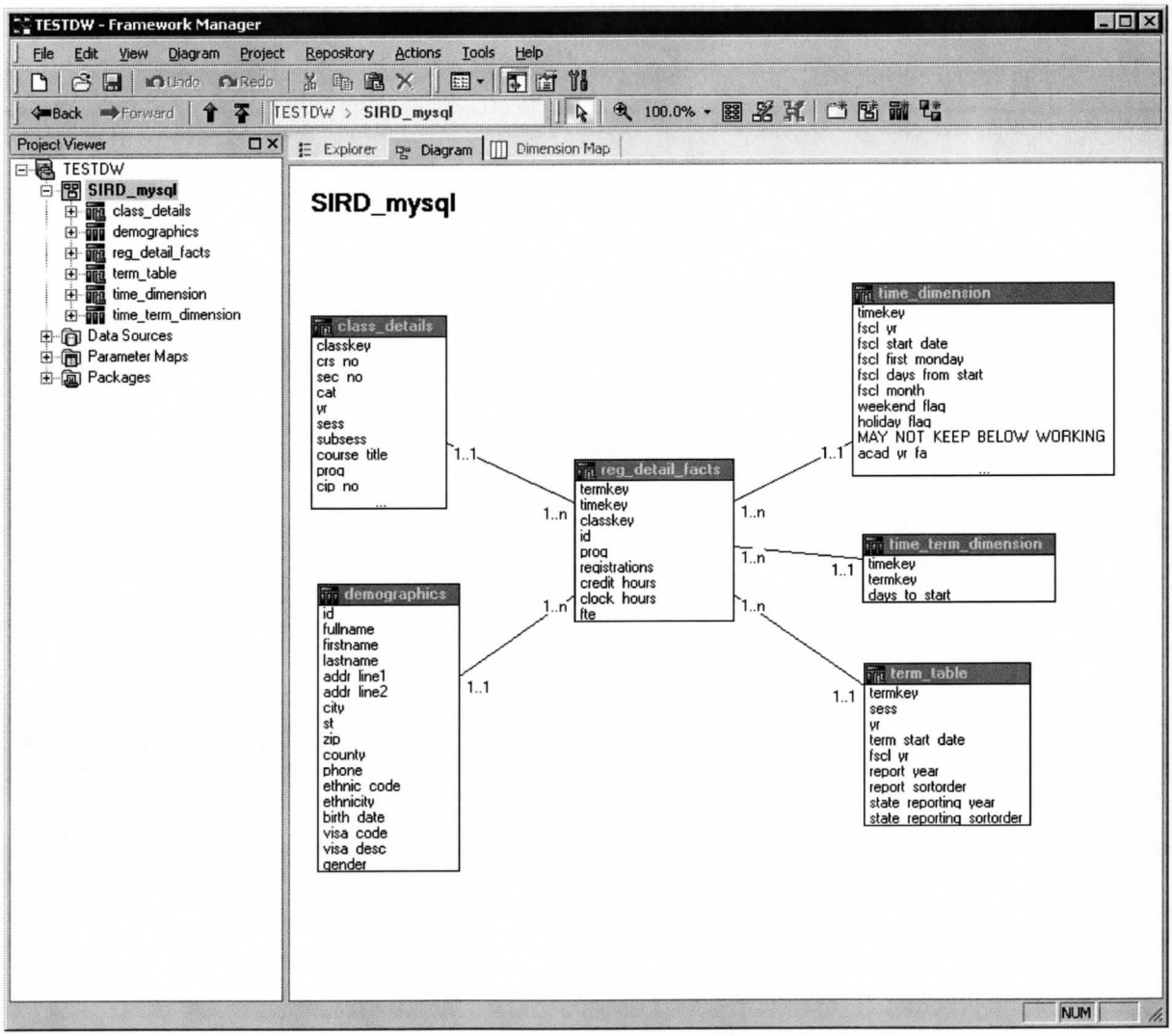


Figure 5.6 Cognos Framework Manager Design

Once the data items have been created and mapped, the information is now available for the Cognos reporting system. With this tool it is now a simple task to create reports that can be used for data analysis. Figure 5.7 shows the reporting studio tool that is used for report generation, it is a straightforward task to create a report. Data elements are simply dragged from the data pane on the left over to the reporting pane on the right side. The report is then run (see figure 5.8) to obtain the information that is desired. The samples shown below in the two figures took less than 5 minutes to create, run, and extract the data for two years worth of information by term.

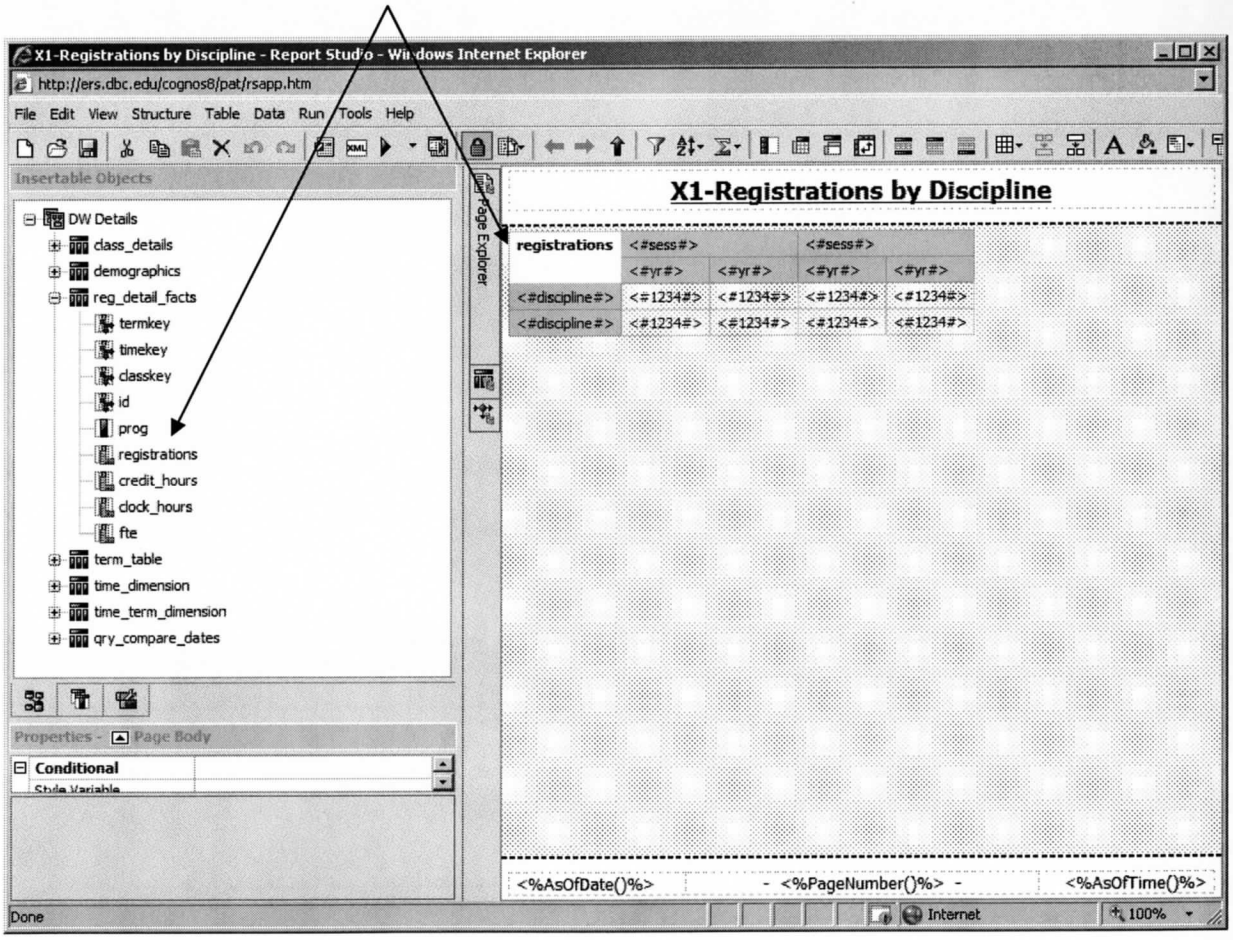
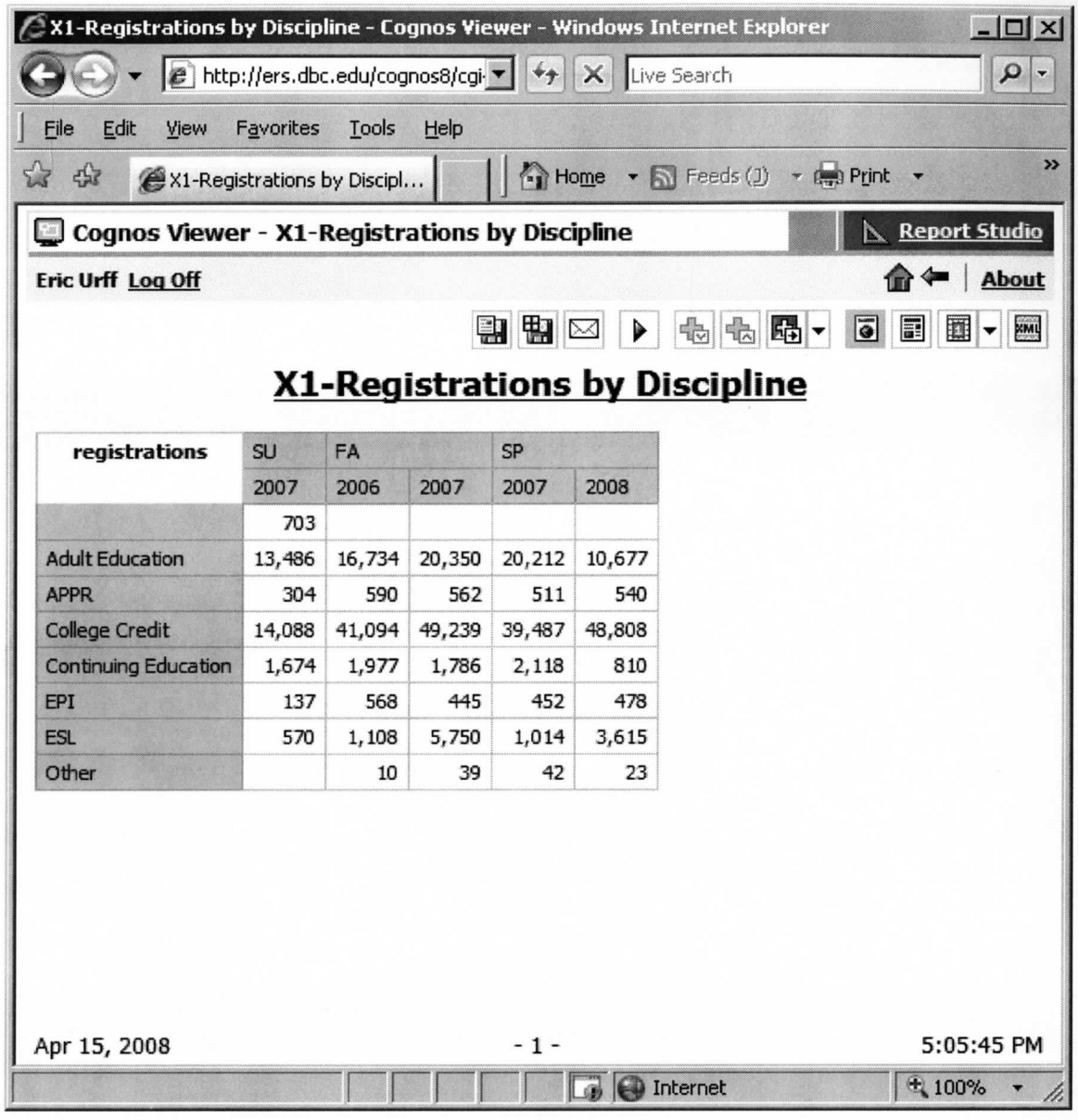


Figure 5.7 Cognos Report Design



registrations	SU	FA		SP	
	2007	2006	2007	2007	2008
	703				
Adult Education	13,486	16,734	20,350	20,212	10,677
APPR	304	590	562	511	540
College Credit	14,088	41,094	49,239	39,487	48,808
Continuing Education	1,674	1,977	1,786	2,118	810
EPI	137	568	445	452	478
ESL	570	1,108	5,750	1,014	3,615
Other		10	39	42	23

Figure 5.8 Cognos Report Results

The report is now available for all users that have permission to run on demand and extract the information that is required. With simple training the Institutional Research department and other authorized areas of the college will be able to manipulate the report and get different sets of results with only a few clicks of the mouse.

CASE STUDY (RESULTS AND DISCUSSION)

One of the primary functions of creating this system was to use the basic tools that were available within the database systems so that it would be easy to replicate and efficient. There was use for a C++ program in order to help generate each of the days in the TIME_DIMENSION table; however this table could have been generated by other means if necessary. By using the Informix SQL tool, and the MYSQL sql statements it is totally within scripts that are run on the HP server for entire extractions and inputs from one system into another. The scripts are automated and run on a continual basis, some portions of the extract are run multiple times throughout the day. DBC does have the luxury of a very powerful HP server and storage area network that is capable of housing all of the necessary data for efficient processing and storage requirements.

Creating a data warehouse of student database by extracting data from a transactional system proved to not be as simple as first planned. Making sure that all of the information that is extracted is correct is the single most important aspect of this project. The data that is used for reporting purposes and analysis in the warehouse must prove to "balance" with the existing data in the day to day transactional system or none of the reports can be relied upon for data analysis. Making sure that the numbers balanced between the systems was the single most difficult factor in the process of creating the warehouse. The totals that are generated for selection and comparison criteria had to match perfectly when checking between the systems; it is unacceptable to have any variance in the data retrievals.

The entire process of extracting and balancing data involved correcting data that may have been in the actual transactional database and was not just "adjusted" during an extraction process in many cases. Courses had invalid department codes that were identified along with

other anomalies. Fortunately for the college data integrity is a very strict function that is not taken lightly and personnel in other departments were willing to help with the correction of the necessary elements to bring data in compliance. This is also due to the fact that the state of Florida requires student data from all of the colleges and universities throughout the state, so being in compliance is mandatory, especially since most of the funding comes from the results of the data that is transmitted to the state reporting system.

The data warehouse now allows the enrollment specialists to continuously monitor and evaluate enrollment statistics on a daily basis in order to compare term data from year to year. The warehouse will be an evolving entity within the college and more tables and data will need to be extracted for more detailed analysis, this will start to include financial data which will be the next phase of this project that has been started.

CONCLUSIONS

The Student Data Repository and Warehouse database that has been created for DBC will be widely used both internally and externally within the college. It is now easier to extract relevant data since there are significantly less tables to join in order to get data out of the system that is necessary for analysis purposes. Information can now be extracted in a matter of seconds so that data can be analyzed. This initial phase of the warehouse is a success and will be used by departments throughout the institution. The initial timeframes have been stretched and were not adhered to, this was mainly to the fact that this was not the only project that I have been involved with during my daily duties.

The future of the warehouse is bright and will be an ongoing process for quite a while. The Spiral Methodology as described in chapter 4 is the most useful technique for this type of project and will continue to spiral outward. There will be more data that will be extracted and added to the warehouse that needs to be incorporated into the current data. Since the college has the Cognos Suite of reporting tools available it will make it easier to bring standard information and create reports that will be able to be used by many individuals throughout the college. The Institutional Research department is using reports to make the data readily available to others without the need to reinvent the information for each area of the college.

During the creation of the warehouse it was necessary to be able to adjust different techniques of extraction and to develop some different routines for actually storing the data. It was actually beneficial to apply some of the techniques that were learned in our courses that helped to apply to the storage elements of the data warehouse. These included how to store data along with what data types were most beneficial for specific reporting purposes.

Indexing the datasets with different attributes and variables also sped up some of the retrieval and extraction routines.

The college had over 45,000 registrations (a registration is counted as each student that is registered in a class) for the fall 2007 semester, this makes the tracking and availability of the information critical to daily operations. This data warehouse is currently being used by the Vice President of Enrollment Services and the Dean of Enrollment Management to obtain registration statistics daily. Departments can obtain statistics to compare registration rates on a term by term basis. The bookstore uses reports to insure that they have ordered enough books so that all students have materials that are necessary. The immediate need of the college will be to create a set of standard reporting elements that will be used for analysis within departments of the college and will not require Institutional Research to be on the call for every report that is requested on a daily basis.

Since the warehouse is contained within a MYSQL database system it has produced the benefit that none of the reporting needs effect the daily transactional system since all of the information is extracted into a totally separate database. This is a big concern of the administration at the college since history has shown that most of the information is requested during the time frame that routinely surrounds the beginning of each term. This time is when the transactional system is being hit upon the most for student registration, billing, and other essential operations of the college.

I will continue to add data to this system and make it available in a format that end users will be able to use and analyze. Different users have different abilities so some data can be made available in raw format that can be extracted and analyzed by individual familiar with database design. Other data will be made available in standard reports with simple

parameters that can be used to generate the information that is requested. It is now possible to extract and compare multiple years of information in a matter of seconds that is not feasible against the transactional database.

Continuing to move forward in the spiral approach will work in the future success of this reporting database. Each portion of the project will add more information that will be able to be reported on and analyzed, some information will be able to stand alone while other bits will be integrated into the existing data. This is one of the most beneficial methods that I learned throughout the entire program, by getting started and continuously moving forward it is possible to produce information and results on a ongoing fashion. Trying to create the entire system at the beginning would delay any information that people at the college could use.

REFERENCES

Accountability, Research & Measurement, MISATFOR (2007) http://www.fldoe.org/arm/cctcmis

Dennis, A., Haley Wixom, B., Tegarden, D. (2005). Systems Analysis and Design with UML Version 2.0. John Wiley & Sons, Inc.

Giovinazzo, W. A. (2000). Object-oriented data warehouse design: Building a star schema. New Jersey: Prentice Hall

Greenfield, L. (2008). The Data Warehousing Information Center. http://www.dwinfocenter.org

Guan, J., Nunez, W., Welsh, J. F., . (2002). Institutional strategy and information support: The role of data warehousing in higher education. Campus - Wide Information Systems, *19*(5), 168-174. Retrieved January 27, 2008, from ABI/INFORM Global database. (Document ID: 277434221).

Parekh, Nilesh (2005). Spiral Model - A New Approach Towards Software Development. Retrieved January 27, 2008, from http://www.buzzle.com/editorials/1-13-2005-64082.asp

APPENDIX A: WORK BREAKDOWN STRUCTURE

1. **Planning**
   - 1.1. Scope
   - 1.2. WBS
   - 1.3. Identify information required
   - 1.4. Select tools
     - 1.4.1. Programming tools
     - 1.4.2. Database
     - 1.4.3. Reporting tools
   - 1.5. Prioritize tasks
   - 1.6. Schedule
     - 1.6.1. Task Timelines
     - 1.6.2. Gantt Chart
   - 1.7. Web server specs
2. **Execution**
   - 2.1. User Input
     - 2.1.1. Define variables to extract
     - 2.1.2. Warehouse elements to maintain
   - 2.2. Transactional Database Information
     - 2.2.1. Identify transactional database fields
     - 2.2.2. Define data to extract
     - 2.2.3. Define any transactional to warehouse filters

2.3. Warehouse Database Design

2.3.1. Database server location

2.3.2. Define Calendar days

2.3.3. Design Tables

2.4. Programming

2.5. Design Package (Cognos catalog)

2.5.1. Framework Manager

2.5.2. Package areas for reporting

2.6. Phase I reports

2.7. Testing

2.8. Training

3. Closing

3.1. Final Project Report

3.2. Presentation

3.3. Lessons Learned

APPENDIX B: GANNT CHART

APPENDIX C: SAMPLE REPORTS

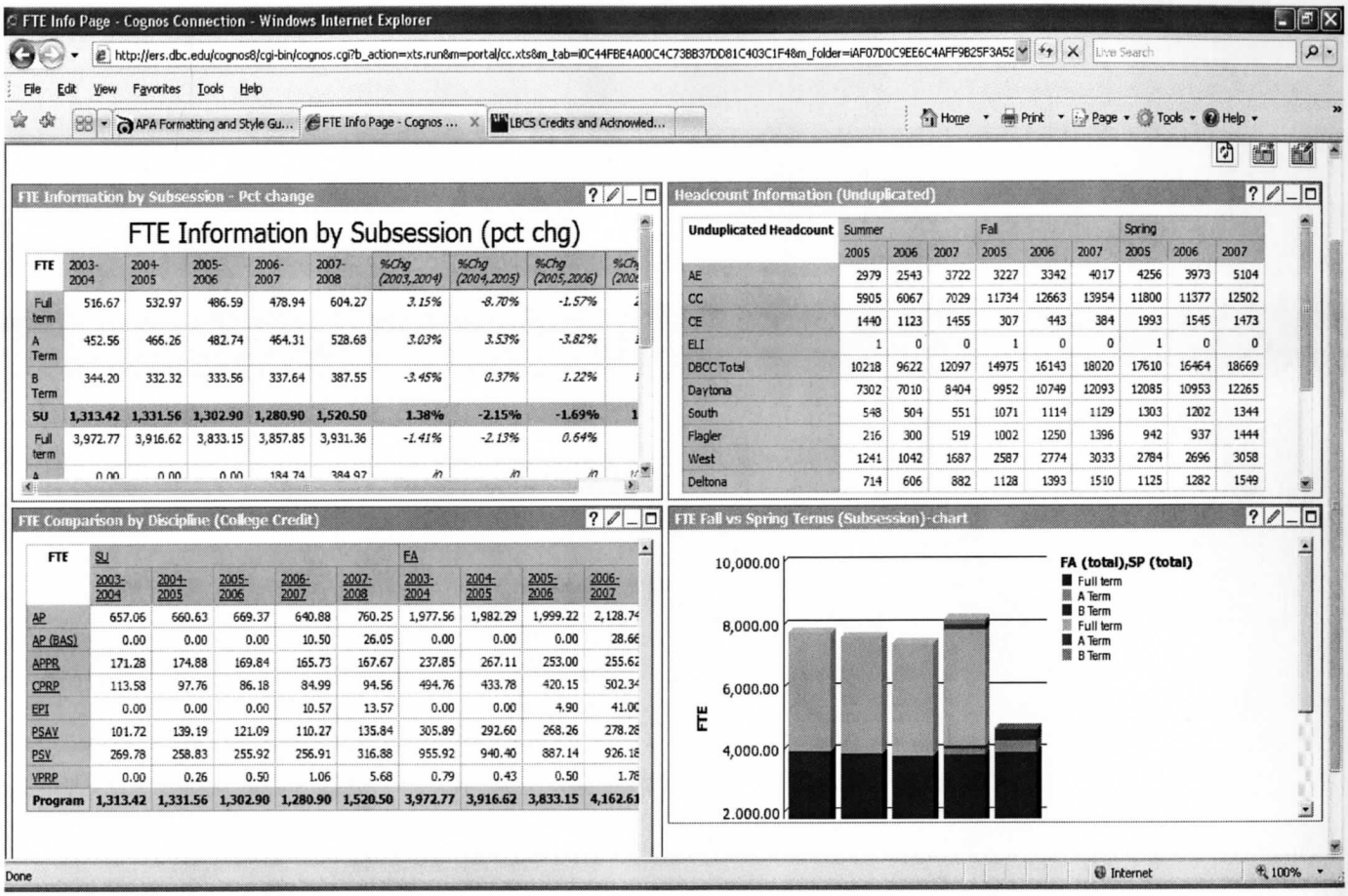
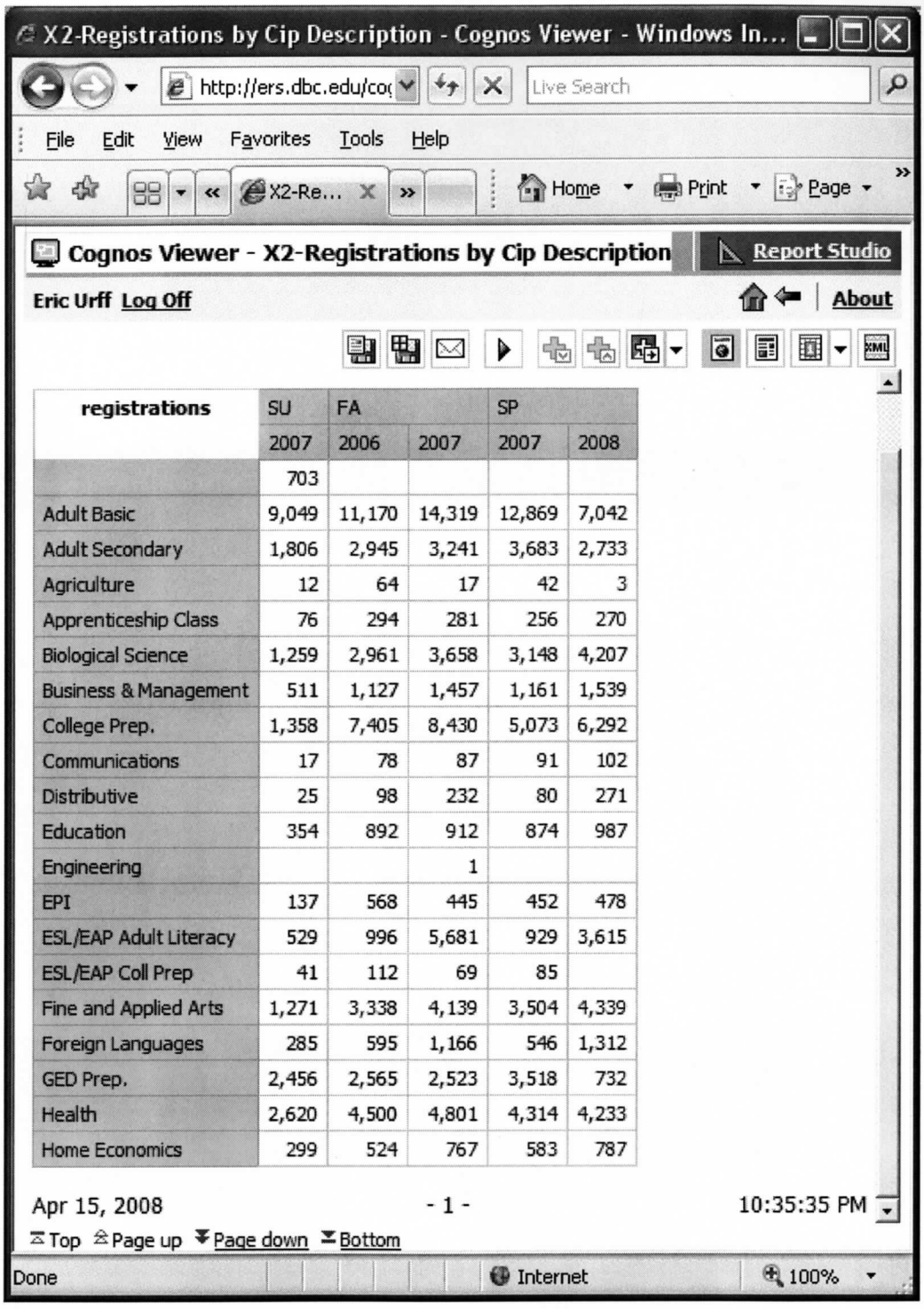
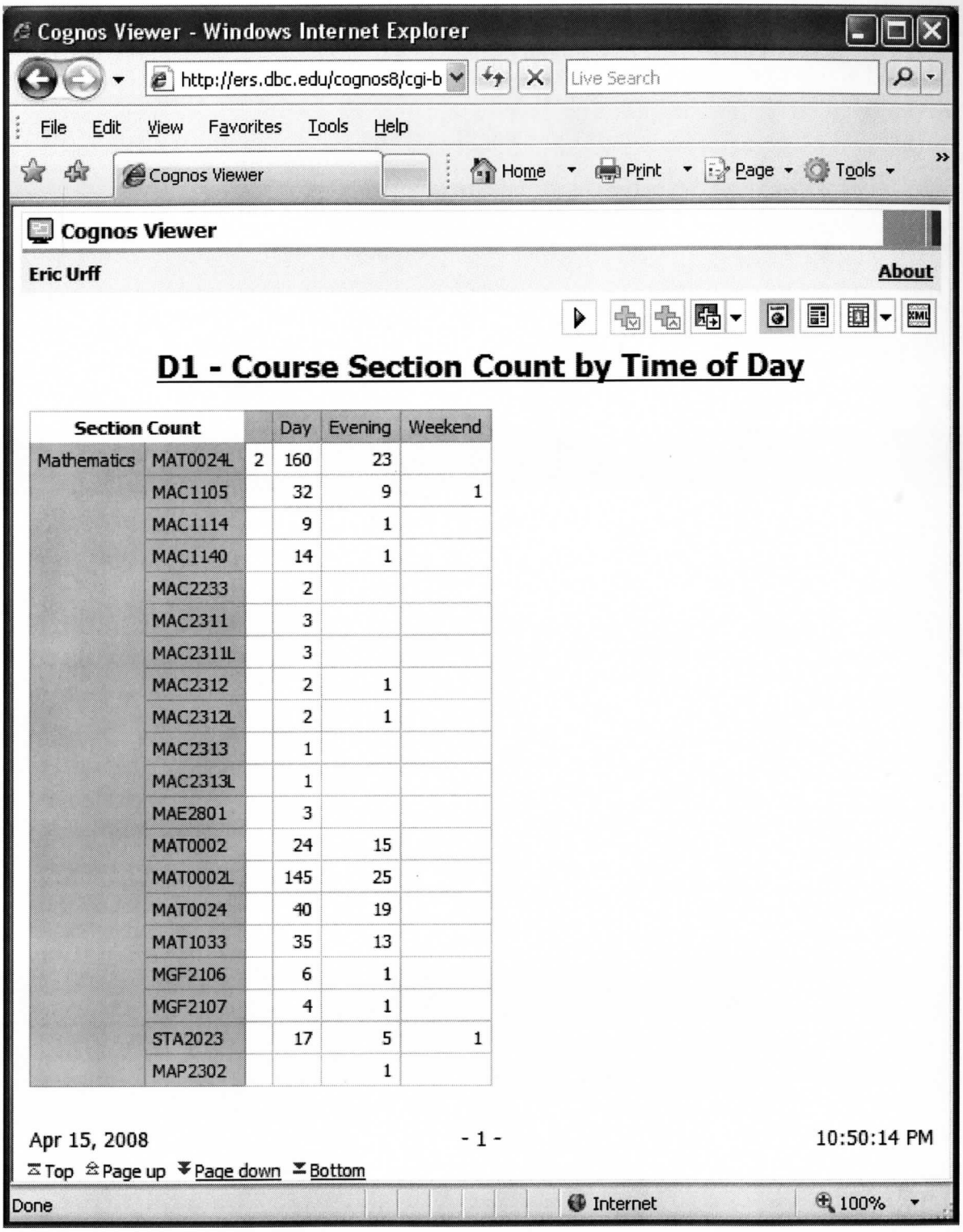


Figure C1: Multiple FTE reports on single page



registrations	SU	FA		SP	
	2007	2006	2007	2007	2008
	703				
Adult Basic	9,049	11,170	14,319	12,869	7,042
Adult Secondary	1,806	2,945	3,241	3,683	2,733
Agriculture	12	64	17	42	3
Apprenticeship Class	76	294	281	256	270
Biological Science	1,259	2,961	3,658	3,148	4,207
Business & Management	511	1,127	1,457	1,161	1,539
College Prep.	1,358	7,405	8,430	5,073	6,292
Communications	17	78	87	91	102
Distributive	25	98	232	80	271
Education	354	892	912	874	987
Engineering			1		
EPI	137	568	445	452	478
ESL/EAP Adult Literacy	529	996	5,681	929	3,615
ESL/EAP Coll Prep	41	112	69	85	
Fine and Applied Arts	1,271	3,338	4,139	3,504	4,339
Foreign Languages	285	595	1,166	546	1,312
GED Prep.	2,456	2,565	2,523	3,518	732
Health	2,620	4,500	4,801	4,314	4,233
Home Economics	299	524	767	583	787

Figure C2: Registrations by CIP code and session, year



Section Count			Day	Evening	Weekend
Mathematics	MAT0024L	2	160	23	
	MAC1105		32	9	1
	MAC1114		9	1	
	MAC1140		14	1	
	MAC2233		2		
	MAC2311		3		
	MAC2311L		3		
	MAC2312		2	1	
	MAC2312L		2	1	
	MAC2313		1		
	MAC2313L		1		
	MAE2801		3		
	MAT0002		24	15	
	MAT0002L		145	25	
	MAT0024		40	19	
	MAT1033		35	13	
	MGF2106		6	1	
	MGF2107		4	1	
	STA2023		17	5	1
	MAP2302			1	

Figure C3: Course Section Counts by time of day

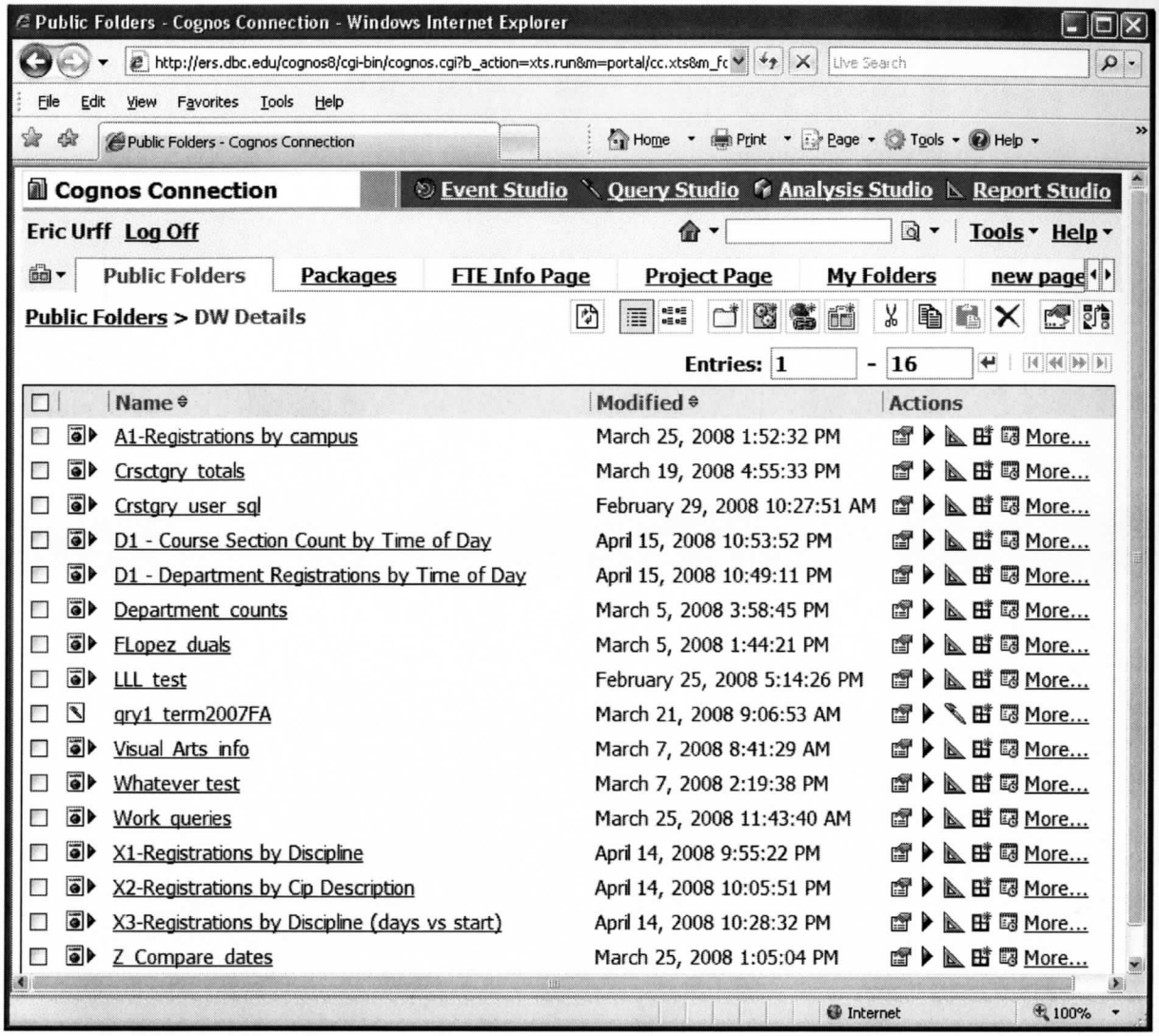


Figure C4: Reporting Menu Structure

APPENDIX D: SYSTEM TECHNICAL DOCUMENTATION

MYSQL Student Information Repository Database – SQL Creation statements

```
/*
Created         12/07/2008
Modified        02/11/2008
Project         Student Information Repository Database
Model
Company
Author          Eric Urff
Version
Database        mySQL 5
*/

Create table REGISTRATION_FACTS (
        termkey Char(6) NOT NULL,
        timekey Date NOT NULL,
        classkey Int NOT NULL,
        registrations Int DEFAULT 0,
        prog Char(4) DEFAULT '',
        credit_hours Double DEFAULT 0.0000000000000000,
        clock_hours Double DEFAULT 0.0000000000000000,
        fte Double DEFAULT 0.00) ENGINE = InnoDB;

Create table CLASS_DETAILS (
        classkey Int NOT NULL,
        crs_no Char(12) NOT NULL DEFAULT '',
        sec_no Char(4) NOT NULL,
```

```
cat Char(4) NOT NULL DEFAULT '',
yr Int NOT NULL,
sess Char(20) NOT NULL,
subsess Char(20) NOT NULL DEFAULT ' ',
course_title Varchar(32),
prog Char(4),
cip_no Char(8),
cip_description Varchar(30),
crsctgry Char(4),
discipline Varchar(25) DEFAULT 'Unknown',
dept Char(4),
dept_descr Varchar(30),
hrs Double DEFAULT 0.0000000000000000,
clock_hrs Double DEFAULT 0.0000000000000000,
fte_type Char(10) DEFAULT 'fte0',
tuit_code Char(4),
fee_code Char(4),
bill_code Char(4),
gordon Int,
fund Char(2),
subfund Char(9),
func Char(5),
crs_major Char(4) DEFAULT '',
crs_major_desc Varchar(30) DEFAULT ' ',
div_id Char(4),
div_desc Varchar(30),
max_reg Int DEFAULT 0,
faculty_id Int DEFAULT 0,
faculty_name Varchar(35) DEFAULT ' ',
campus_code Char(4),
campus_name Varchar(20),
```

```
        im Char(20),
        im_descr Varchar(25),
        dayeve_wkend Varchar(10) DEFAULT ' ',
        days Char(7) DEFAULT ' ',
        start_time Int DEFAULT 0,
        UNIQUE (classkey),
 Primary Key (classkey)) ENGINE = InnoDB;

Create table DEPT_TABLE (
        dept Char(4) NOT NULL,
        dept_descr Varchar(30) NOT NULL,
        dept_chair_id Int NOT NULL DEFAULT 0,
        dept_chair_name Varchar(35),
        div_id Char(4),
        UNIQUE (dept),
 Primary Key (dept)) ENGINE = MyISAM;

Create table DIV_TABLE (
        div_id Char(4) NOT NULL,
        div_descr Varchar(30) NOT NULL,
        div_dean_id Int NOT NULL DEFAULT 0,
        div_dean_name Varchar(35),
        UNIQUE (div_id),
 Primary Key (div_id)) ENGINE = MyISAM;

Create table TIME_DIMENSION (
        timekey Date NOT NULL,
        fscl_yr Char(4) NOT NULL,
        fscl_start_date Date NOT NULL,
        fscl_first_monday Date NOT NULL DEFAULT '2007-06-30',
        fscl_days_from_start Int NOT NULL DEFAULT 0,
```

```
        fscl_month Varchar(10) NOT NULL,
        weekend_flag Int NOT NULL DEFAULT 0,
        holiday_flag Int NOT NULL DEFAULT 0,
        UNIQUE (timekey),
 Primary Key (timekey)) ENGINE = InnoDB;

Create table TERM_TABLE (
        termkey Char(6) NOT NULL,
        sess Char(4) NOT NULL,
        yr Int NOT NULL DEFAULT 0,
        term_start_date Date NOT NULL,
        fscl_yr Char(4) NOT NULL,
        report_year Char(10) NOT NULL DEFAULT '1990-1991',
        report_sortorder Int NOT NULL,
        state_reporting_year Int NOT NULL DEFAULT 1990,
        state_reporting_sortorder Int NOT NULL DEFAULT 1,
        UNIQUE (termkey),
 Primary Key (termkey)) ENGINE = InnoDB;

Create table REG_DETAIL_FACTS (
        termkey Char(6) NOT NULL,
        timekey Date NOT NULL,
        classkey Int NOT NULL,
        id Int NOT NULL,
        prog Char(4) DEFAULT '',
        registrations Int,
        credit_hours Double DEFAULT 0.0000000000000000,
        clock_hours Double DEFAULT 0.0000000000000000,
        fte Double,
 Primary Key (termkey,timekey,classkey,id)) ENGINE = InnoDB;
```

```
Create table DEMOGRAPHICS (
        id Int NOT NULL,
        fullname Char(20),
        firstname Varchar(20),
        lastname Varchar(20),
        addr_line1 Varchar(25),
        addr_line2 Varchar(25),
        city Char(20),
        st Char(2),
        zip Char(10),
        county Char(20),
        phone Varchar(20),
        ethnic_code Char(4),
        ethnicity Varchar(20),
        birth_date Date,
        visa_code Char(4),
        visa_desc Varchar(20),
        gender Char(20),
        UNIQUE (id),
 Primary Key (id)) ENGINE = InnoDB;

Create table TIME_TERM_DIMENSION (
        timekey Date NOT NULL,
        termkey Char(6) NOT NULL,
        days_to_start Int DEFAULT 0) ENGINE = InnoDB;


Create UNIQUE Index termtbl_idx1 ON TERM_TABLE (sess,yr);
Create Index termtbl_idx2 ON TERM_TABLE (report_year,state_reporting_year,sess,yr);
Create Index regdtl_idx1 ON REG_DETAIL_FACTS (classkey);
Create Index regdtl_idx2 ON REG_DETAIL_FACTS (timekey);
```

```
Create Index regdtl_idx3 ON REG_DETAIL_FACTS (termkey);
Create UNIQUE Index tt_idx1 ON TIME_TERM_DIMENSION (termkey,days_to_start);
```

CLASS_DETAILS Extraction/Load

```
unload to dat_classdtls.out

select

 d.seckey as classkey

, d.crs_no, d.sec_no, d.cat, d.yr, d.sess, s.subsess

, c.title1 as course_title

, c.prog, c.cip_no, cip.txt, c.crsctgry

, c.dept, dept.txt as dept_descr

, s.hrs, s.clock_hrs

, s.tuit_code, s.fee_code, s.bill_code

, c.gordon, c.fund, c.subfund, c.function as func

, c.rpt_major as crs_major, maj.title as crs_major_desc

, dept.div as div, 'div_desc' as div_desc

, s1.mtg_no

 from dwseckey d, sec_rec s, crs_rec c

 , outer cip_table cip

 , outer dept_table dept

 , outer rpt_major_table maj

 , outer secmtgmain_vw s1


  where s.crs_no = d.crs_no

    and s.cat    = d.cat

    and s.yr     = d.yr

    and s.sess   = d.sess
```

```
  and s.sec_no = d.sec_no

and c.crs_no = d.crs_no

and c.cat    = d.cat

  and cip.cip_no = c.cip_no

and dept.dept = c.dept

  and c.rpt_major = maj.rpt_major

and s1.crs_no = d.crs_no

and s1.cat    = d.cat

and s1.yr     = d.yr

and s1.sess   = d.sess

and s1.sec_no = d.sec_no
```

Load statement for CLASS_DETAILS into MYSQL

- load data infile '/work/eu/dat_classdtls.out' REPLACE into table class_details fields

Sample Data: CLASS_DETAILS extract

46317|0104340|01|CC02|2001|FA| |Drawing I|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.8399999999999999|85.2 00000000000003|fte900| | | |0|10|111320200|00000|4001|DBCC Adult High School|AE|div_desc|150|221776|Palmieri, Dora A|

46318|0104340|02|CC02|2001|FA| |Drawing I|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.77|83.099999999999994 |fte900| | | |0|10|111320200|00000|4001|DBCC Adult High School|AE|div_desc|150|220753|Wight, Julie E.|

46319|0104340|04|CC02|2001|FA| |Drawing I|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.8399999999999999|85.2 00000000000003|fte900| | | |0|10|111320200|00000|4001|DBCC Adult High School|AE|div_desc|150|92932|Volkmann, Audrey S.|

46320|0104350|01|CC02|2001|FA| |Drawing II|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.8399999999999999|85. 200000000000003|fte900| | | |0|10|111320200|00000|4001|DBCC Adult High School|AE|div_desc|150|221776|Palmieri, Dora A|

46321|0104350|02|CC02|2001|FA| |Drawing II|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.77|83.09999999999999 4|fte900| | | |0|10|111320200|00000|4001|DBCC Adult High School|AE|div_desc|150|220753|Wight, Julie E.|

46322|0104350|04|CC02|2001|FA| |Drawing II|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.8399999999999999|85.
200000000000003|fte900| | | |0|10|111320200|00000|4001|DBCC Adult High School|AE|div_desc|150|92932|Volkmann, Audrey S.|

46323|0200300|01|CC02|2001|FA| |Intro to Computers|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.839999999999
9999|85.200000000000003|fte900| | | |0|10|111320200|00000|4001|DBCC Adult High School|AE|div_desc|150|160396|Godfrey, Rick|

46324|0200300|03|CC02|2001|FA| |Intro to Computers|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.839999999999
9999|85.200000000000003|fte900| | | |0|10|111320200|00000|4001|DBCC Adult High School|AE|div_desc|150|472644|Tse, Kentsun|

46325|0200300|04|CC02|2001|FA| |Intro to Computers|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.839999999999
9999|85.200000000000003|fte900| | | |0|10|111320200|00000|4001|DBCC Adult High School|AE|div_desc|75|287652|Sledge, Elton S.|

46326|0200310|01|CC02|2001|FA| |Comp Applications I|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.83999999999
99999|85.200000000000003|fte900| | | |0|10|111320200| |4001|DBCC Adult High School|AE|div_desc|150|160396|Godfrey, Rick|

46327|0200310|03|CC02|2001|FA| |Comp Applications I|AE|13202|Adult Secondary|AHS|Adult Education|AHS|Adult High School|2.83999999999
99999|85.200000000000003|fte900| | | |0|10|111320200| |4001|DBCC Adult High School|AE|div_desc|150|287652|Sledge, Elton S.|

Part 2 – CLASS_DETAILS – load/merge

```
select
'update class_details '
|| "set dayeve_wkend='" ||
    case  when days like ('S------') then 'Weekend'
      when days like ('------S') then 'Weekend'
      when beg_tm <= 1700       then 'Day'
      when beg_tm >  1700       then "Evening"
       else 'Unknown'
 end || "', "
|| "im='" || m.im || "', im_descr='" || trim(imt.descr)
|| "', campus_code='" || m.campus || "', campus_name='" || trim(camp.campus_name) || """
|| ", days='" || m.days || """
|| "', start_time=" || m.beg_tm
, " where classkey= " || d.seckey
 from dwseckey d, sec_rec s
  ,  secmtgmain_vw s1
  ,  mtg_rec m
  ,  im_table imt
  , outer dbcampus_rec camp
 where s.crs_no = d.crs_no
    and s.cat    = d.cat
    and s.yr     = d.yr
    and s.sess   = d.sess
    and s.sec_no = d.sec_no
   and s1.crs_no = d.crs_no
   and s1.cat    = d.cat
   and s1.yr     = d.yr
   and s1.sess   = d.sess
   and s1.sec_no = d.sec_no
    and m.mtg_no = s1.mtg_no
```

and m.im = imt.im

and camp.campus_num = m.campus

Command Line interface Load into MYSQL

- mysql sird --delimiter="|" < dat_class2.out

Sample Data: Class_details data merge (part 2):

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture', campus_code='0001', campus_name='Daytona', days ='-MTWRF-', start_time=1100 where classkey= 46317 |

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture', campus_code='0002', campus_name='New Smyrna Bea ch', days='-MTWRF-', start_time=1200 where classkey= 46319 |

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture', campus_code='0001', campus_name='Daytona', days ='-MTWRF-', start_time=1100 where classkey= 46320 |

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture', campus_code='0002', campus_name='New Smyrna Bea ch', days='-MTWRF-', start_time=1200 where classkey= 46322 |

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture', campus_code='0001', campus_name='Daytona', days ='-MTWRF-', start_time=800 where classkey= 46323 |

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture', campus_code='0002', campus_name='New Smyrna Bea ch', days='-MTWRF-', start_time=800 where classkey= 46324 |

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture',
campus_code='0005', campus_name='Four Townes',
days='-MTWRF-', start_time=800 where classkey= 46325 |

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture',
campus_code='0001', campus_name='Daytona', days
='-MTWRF-', start_time=800 where classkey= 46326 |

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture',
campus_code='0005', campus_name='Four Townes',
days='-MTWRF-', start_time=1000 where classkey= 46327 |

update class_details set dayeve_wkend='Day', im='LE', im_descr='Lecture',
campus_code='0002', campus_name='New Smyrna Bea
ch', days='-MTWRF-', start_time=900 where classkey= 46328 |

TERM_TABLE extraction and load

```
unload to termtbl.out
select  yr||sess  as termkey
, sess, yr, to_char(beg_date, '%Y-%d'     )
, acyr
,case when sess='FA' then yr   || "-" || yr+1
 else              yr-1 || "-" || yr
 end  as report_year
,case when sess='FA' then 1
    when sess='SP' then 2
    else              3
 end as report_sortorder
,case when sess='FA' then yr||"
     when sess='SP' then yr-1||"
     when sess='SU' then yr||"
    else              '0'
 end as state_reporting_year
,case when sess='FA' then 2
     when sess='SP' then 3
     when sess='SU' then 1
    else              0
 end as state_reporting_sortorder
 from acad_cal_rec
 where prog='CC' and subsess=' '
  and yr > 2000
  and acyr >= '0102'
order by state_reporting_year, state_reporting_sortorder
```

Command Line interface Load TERM_TABLE into MYSQL

- load data infile '/work/eu/dat_termtbl.out' REPLACE into table term_table fields terminated by '|';

Sample Data: Term_table

2001FA|FA|2001|2001-08-20|0102|2001-2002|1|2001|2|

2002SP|SP|2002|2002-01-07|0102|2001-2002|2|2001|3|

2002SU|SU|2002|2002-05-08|0102|2001-2002|3|2002|1|

2002FA|FA|2002|2002-08-19|0203|2002-2003|1|2002|2|

2003SP|SP|2003|2003-01-06|0203|2002-2003|2|2002|3|

2003SU|SU|2003|2003-05-07|0203|2002-2003|3|2003|1|

2003FA|FA|2003|2003-08-25|0304|2003-2004|1|2003|2|

2004SP|SP|2004|2004-01-12|0304|2003-2004|2|2003|3|

2004SU|SU|2004|2004-05-12|0304|2003-2004|3|2004|1|

2004FA|FA|2004|2004-08-23|0405|2004-2005|1|2004|2|

2005SP|SP|2005|2005-01-10|0405|2004-2005|2|2004|3|

2005SU|SU|2005|2005-05-11|0405|2004-2005|3|2005|1|

2005FA|FA|2005|2005-08-22|0506|2005-2006|1|2005|2|

2006SP|SP|2006|2006-01-09|0506|2005-2006|2|2005|3|

2006SU|SU|2006|2006-05-10|0506|2005-2006|3|2006|1|

2006FA|FA|2006|2006-08-23|0607|2006-2007|1|2006|2|

2007SP|SP|2007|2007-01-08|0607|2006-2007|2|2006|3|

2007SU|SU|2007|2007-05-09|0607|2006-2007|3|2007|1|

2007FA|FA|2007|2007-08-22|0708|2007-2008|1|2007|2|

2008SP|SP|2008|2008-01-14|0708|2007-2008|2|2007|3|

2008SU|SU|2008|2008-05-14|0708|2007-2008|3|2008|1|

2008FA|FA|2008|2008-08-25|0809|2008-2009|1|2008|2|

2009SP|SP|2009|2009-01-12|0809|2008-2009|2|2008|3|

2009SU|SU|2009|2009-05-13|0809|2008-2009|3|2009|1|

2009FA|FA|2009|2009-08-17|0910|2009-2010|1|2009|2|

2010SP|SP|2010|2010-01-04|0910|2009-2010|2|2009|3|

2010FA|FA|2010|2010-08-18|1011|2010-2011|1|2010|2|

2011SP|SP|2011|2011-01-05|1011|2010-2011|2|2010|3|

2011FA|FA|2011|2011-08-18|1112|2011-2012|1|2011|2|

2012SP|SP|2012|2012-01-05|1112|2011-2012|2|2011|3|

2012FA|FA|2012|2012-08-18|1213|2012-2013|1|2012|2|

2013SP|SP|2013|2013-01-05|1213|2012-2013|2|2012|3|

2013FA|FA|2013|2013-08-18|1314|2013-2014|1|2013|2|

2014SP|SP|2014|2014-01-05|1314|2013-2014|2|2013|3|

BUILD_TIME_TERM Procedure in MYSQL

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `sird`.`build_time_term`$$
CREATE DEFINER=`urff`@`10.14.2.36` PROCEDURE `build_time_term`( )
BEGIN
 DECLARE dt CHAR(10);
 DECLARE trmkey CHAR(6);
 DECLARE stdt CHAR(10);
 DECLARE i, j, k int;
 DECLARE dtdiff int DEFAULT 180;
 DECLARE done INT DEFAULT 0;
 DECLARE cur1 CURSOR FOR SELECT termkey FROM term_table;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
 OPEN cur1;
 REPEAT
  FETCH cur1 INTO trmkey;
  SET dtdiff = 180;
  SELECT  date_add(term_start_date, interval dtdiff day) into stdt
   from term_table
   where termkey = trmkey;
  WHILE dtdiff > -180 DO
   REPLACE into  time_term_dimension  values (stdt, trmkey, dtdiff);
   SET stdt = date_sub(stdt, interval 1 day);
   SET dtdiff = dtdiff - 1;
  END WHILE;
 UNTIL done END REPEAT;
 CLOSE cur1;
END$$
DELIMITER ;
```

To run the above procedure in MYSQL

- call build_time_term( )

Note: This will generate a record into the time_term_table for each term that is in the term_table. It will go from 180 days prior to and 180 days after the start of term date for each term.

INFORMIX SQL to check for any adjusted reg_rec

```
select r.*
from reg_rec r, cw_rec c
 where r.cw_no = c.cw_no
 and c.sess='FA' and
```

INFORMIX PROCEDURE – to FIX out of sequence registration records

```
drop procedure dwfixregrec;
create procedure dwfixregrec (
    in_sess char(4),
    in_yr int)
  returning
    char(50);
  DEFINE xrcwno   int;
  DEFINE xsysdate date;
  DEFINE xtm      int;
  DEFINE xrstat   char(2);
  DEFINE xcstat   char(2);
  DEFINE xreason  char(4);
  DEFINE xcprog   char(4);
  DEFINE xruserid int;
  DEFINE xregval  int;
  DEFINE xcregval int;
  DEFINE result   char(50);
  DEFINE recsadjusted  int;

  DEFINE wcwno    int;
  DEFINE wtotval  int;
  DEFINE wupdsw   int;
  LET result      = "";
```

```
 LET wtotval     = 0;
 LET wcwno       = 0;
 LET recsadjusted = 0;
FOREACH
 select r.cw_no, sys_date, tm,r.stat , c.stat, r.reason, c.prog, r.user_id
  , case  when r.user_id < 0       then  0
    when r.stat in ('r', 'R') then  1
    when r.stat in ('d', 'D', 'x', 'X') then -1
    else 0
   end as regval
  , case  when c.stat in ('r', 'R', 'w', 'W') then  1
        when c.stat in ('d', 'D', 'x', 'X') then -1
        else 0
   end as cregval
 into xrcwno, xsysdate, xtm, xrstat, xcstat, xreason, xcprog, xruserid, xregval, xcregval
from reg_rec r, cw_rec c
where r.cw_no = c.cw_no
 and   r.stat in ('R', 'r', 'D', 'd', 'x', 'X')
 and   r.cw_no in
 (
  select r.cw_no
   from reg_rec r, cw_rec c
   where r.cw_no = c.cw_no
    and c.sess = in_sess
    and c.yr   = in_yr
    and c.stat in ('R', 'r', 'W', 'w','D','d')
   group by 1 having
    sum(case when r.user_id < 0       then  0
          when r.stat in ('r', 'R') then  1
          when r.stat in ('d', 'D', 'x', 'X') then -1
        else 0
```

```
        end)
      <>
       avg(case   when c.stat in ('r', 'R', 'W', 'w') then  1
                  else 0
          end)
   )
  order by 1,2,3

{ Initialize variables for processing, when the cw_no changes }
  IF (xrcwno != wcwno) THEN
   LET wcwno   = xrcwno;
   LET wtotval = 0;
   LET wupdsw  = 1;
  END IF

  IF (xregval > 0 AND wupdsw = 1) THEN {A valid possiblity }
   LET wupdsw  = 0;                  { Can no longer add a Registration value}
   LET wtotval = wtotval + xregval;
   CONTINUE FOREACH;
  END IF
  {*** Just hold a temporary reset of xuserid in case we need it below ***}
  IF (xruserid > 0) THEN
      LET xruserid = -1 * xruserid;
  ELSE LET xruserid = -1;
  END IF
{ ************************************************************
 ** If a registration value is about to be added and the record
 ** has already been adjusted then we need to reset the userid
 ** so that the value will not be counted
 ************************************************************
}
```

```
  IF (xregval > 0 AND wupdsw = 0) THEN    {** Invalid condition so reset the userid **}
   UPDATE reg_rec set user_id = xruserid
    WHERE cw_no = xrcwno
      and stat  = xrstat;
   LET recsadjusted = recsadjusted + 1;
  END IF

  {**** Do not allow a drop value to be allowed if total values do not coincide ****}
  IF (xregval < 0 AND (wtotval < 1 OR xcregval > 0) ) THEN
   UPDATE reg_rec set user_id = xruserid
    WHERE cw_no = xrcwno
      and stat  = xrstat;
   LET recsadjusted = recsadjusted + 1;
  END IF
 END FOREACH;
 { RETURN trim(result); }
 RETURN "Records Cleaned: " || recsadjusted;
end procedure
```

*** **End Informix Procedure** ***

Registration Detail Facts – Extraction Routine

```
select r.cw_no
,  case when tm < 500 then sys_date -1
          else                    sys_date
   end as sys_date
    , sum (case  when r.user_id < 0 then  0
      when r.stat in ('r', 'R') then  1
      when r.stat in ('d', 'D', 'x', 'X') then -1
      else 0
     end) as regval

   from reg_rec r, cw_rec c
   where r.cw_no = c.cw_no
 and c.sess='SP' and c.yr=2008
group by 1, 2
 into temp temp_dw1;

unload to dat_regdtls.out
select
  trim(c.yr || c.sess) as termkey
, to_char(sys_date, '%Y-%m-%d') as timekey
, d.seckey as classkey, c.id , c.prog
, dw.regval as registrations
, (dw.regval * hrs) as credit_hours
, (dw.regval * clock_hrs) as clock_hours
 from temp_dw1 dw, cw_rec c, dwseckey d
  where c.cw_no = dw.cw_no  and c.crs_no = d.crs_no
   and c.cat  = d.cat and c.yr  = d.yr   and c.sess   = d.sess  and c.sec    = d.sec_no
   and dw.regval <> 0
```

Sample Data: Registration Details Facts Table

2008SP|2007-11-04|136611|311294|CC|1.0|4.0|60.0|

2008SP|2007-11-04|136612|645681|CC|1.0|4.0|60.0|

2008SP|2007-11-04|136616|197237|CC|1.0|4.0|60.0|

2008SP|2007-11-04|136618|648643|CC|1.0|4.0|60.0|

2008SP|2007-11-04|136619|641433|CC|1.0|4.0|60.0|

2008SP|2007-11-04|136627|538813|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136628|240807|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136701|197055|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136701|212099|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136701|577417|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136703|636154|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136713|623358|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136616|197237|CC|1.0|4.0|60.0|

2008SP|2007-11-04|136618|648643|CC|1.0|4.0|60.0|

2008SP|2007-11-04|136619|641433|CC|1.0|4.0|60.0|

2008SP|2007-11-04|136627|538813|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136628|240807|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136701|197055|CC|1.0|3.0|45.0|

2008SP|2007-11-04|136701|212099|CC|1.0|3.0|45.0|